UNIVERSIDAD CARLOS III DE MADRID

Departamento de Ingeniería Telemática

Proyecto Fin de Carrera

"JAVA'S PURGATORY": DESARROLLO DE UN JUEGO EDUCATIVO EN UNITY3D.

Autor: David Gómez Durán
Tutor proyecto: Dr. José Jesús García Rueda

Leganés, Julio de 2013

PROYECTO FIN DE CARRERA

Departamento de Ingeniería Telemática

Universidad Carlos III de Madrid

Título: “JAVA’S PURGATORY”: Desarrollo de un juego educativo en Unity3D

Autor: David Gómez Durán

Tutor del proyecto: Dr. José Jesús García Rueda

EL TRIBUNAL

PRESIDENTE: _____

SECRETARIO: _____

VOCAL: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 2013 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

PRESIDENTE SECRETARIO VOCAL

RESUMEN

Los juegos han estado siempre cerca de nosotros, han servido de maestro mudo a través de los tiempos, y rara vez se olvidan las reglas o sus principios. Por eso ““JAVA’S PURGATORY”: Desarrollo de un juego educativo en Unity3D ” es una guía para todos aquellos que continúan en el mundo de la programación, en la que se propone un recorrido por los siete pecados capitales que tuvo que pasar Dante para poder alcanzar la limpieza de su alma, siendo cada uno de ellos un tema distinto a tratar. Se coge como referencia La Divina Comedia de Dante Alighieri, en la que se presenta al alumno Dante, y su maestro Virgilio. Durante el viaje, se han de limpiar los pecados y aprender de lo que cada personaje ha de contar al viajero, para llegar al final del viaje y poder abrazar a su amada Beatriz. La similitud con el proyecto, es que Dante es un alumno al que todavía le queda la asignatura de Programación de sistemas de primero, y Virgilio es el profesor pedagógico que le guía para superarla.

El principal objetivo del proyecto, es poder aprender los principios básicos en cuanto a la orientación a objetos, acercar a los jugadores a un mundo quizá desconocido para ellos y facilitar el aprendizaje de una asignatura, que en la mayoría de los casos no se ha visto durante bachillerato.

Así mismo, se hace uso de la herramienta para la creación de juegos Unity3D, que nos permite crear mundos, escenarios o todo lo que esté en nuestra mano, con único tope, nuestra imaginación.

ABSTRACT

Games have always been close to us, have served as dumb teacher through the ages, and rarely forget the rules or principles. So “JAVA’S PURGATORY”: Development of an educational game in Unity3D”, is a guide for those who continue in the world of programming, in proposing a tour of the seven deadly sins that Dante had to go to reach his soul cleansing, being each a different subject to be treated. It takes as reference the Divine Comedy of Dante Alighieri, in which the student presents Dante, and his teacher Virgil. During the trip, you have to clean the sins and learn from what each character has to show the traveler, to reach the end of the trip and to embrace his beloved Beatrice. The similarity with the project, is that Dante is a student who still has the first programming course, and Virgil is the teacher who guides you to overcome it.

The main objective of the project is to learn the basics about object orientation, bring players into a world unknown to them and may facilitate the learning of a subject, which in most cases has not been see during high school.

Likewise, the tool uses to create Unity3D games, which allows us to create worlds, scenarios or everything in our power, with only one stop, our imagination.

AGRADECIMIENTOS

Hace mucho tiempo, escuché una frase que dijo alguien una vez:

“Antes de morir, los seres humanos deberíamos: plantar un árbol, escribir un libro y tener un hijo”.

Analizando por que etapa de mi vida voy puedo afirmar, que he plantado un árbol, un melocotonero que da frutos, ya he dejado una semilla para mi familia tanto para el presente, como para el futuro; en cuanto a lo de escribir un libro, durante mi vida, la que llevo vivida, he podido escribir no solo uno sino varios libros de muchas cosas, con cada capítulo distinto al anterior y cada vez con mayor información que aportar a los que lo leyesen, pero con este trabajo pretendo ayudar a los que empiezan en un mundo cuanto menos complicado y lleno de baches, pero también repleto de satisfacción, al ver que lo que uno ha ideado y dado forma, cobra vida de forma en un principio inimaginable....y con respecto a tener un hijo.....eso es otro capitulo de mi vida,jeje.

Ante todo lo antes expuesto, debo agradecer el esfuerzo que ha supuesto y aún supone para mis padres, Jose María y Milagros, el haber conseguido que estudie una carrera y ser alguien de provecho, como diría mi familia, sobre todo mi tío Juan Antonio. A mi hermano, por esas comidas de cabeza, y momentos de idea feliz que tantos problemas nos han quitado de la tediosa vida del estudiante, Fernando sin esos momentos la vida seria muy aburrida. A mis compañeros de carrera, en especial mención a Gema, la cual aún a día de hoy sigue soportando mis vueltas de todo, para luego tener que darme la razón aunque no la tenga. Y a mis amigos de Los Molinos, los cuales han estado siempre en el mejunje de todo este trabajo.

Pero una especial mención se le debe dar a Ana, la mujer que ha compartido y comparte su vida con la mía, y la cual espero que sea la que me ayude a poder concluir el último capítulo que debería hacer antes de terminar el libro de mi vida.

A ella, a mis padres y a mi hermano, está dedicado este trabajo. Gracias.

“Los dos días más importantes de la vida de cada persona son: El día que naces, y el día que encuentras la razón por la que naciste.”

INDICE

CAPÍTULO 1

INTRODUCCIÓN

1.1. Introducción al proyecto.

En el siguiente proyecto se va a tratar lo que se denomina, la Programación Orientación a Objetos , la cual se denominará a partir de este momento como P.O.O. .

El trabajo que se muestra es un juego en primera persona, que transporta al usuario a distintas escenas, ambientadas todas ellas por los pecados del purgatorio, por los que tuvo que pasar Dante para su propia redención. En cada escena se trata un tema distinto relacionado con la P.O.O., y al final de cada una de ellas, se ponen a prueba los conocimientos que el alumno ha adquirido, con una breve pregunta sobre el tema tratado.

El juego está realizado con el programa Unity3D. Esta herramienta es usada para la creación de mundos, espacios virtuales, o simplemente escenarios de la vida cotidiana.

Empresa afincada en San Francisco (EE.UU.), da soporte a creadores Freelance, o a empresas dedicadas específicamente a la creación de juegos, como pueden ser: de carreras, RPG(Role Play Games), educativos, aventuras, shooter, etc., pudiendo crearse en primera o tercera persona, y haciendo uso de un motor gráfico sencillo. [1]

Los lenguajes que admite esta herramienta son: JavaScript[2], C# Script y Boo Script (siendo este último una mezcla entre Python y C#), aunque los dos primeros están entre los más comunes a la hora de programar con ellos, puesto que son lenguajes más conocidos, y por tanto más depurados frente a los posibles errores que se dan durante la creación de programas.

También es un programa que trabaja con la mayoría de los sistemas operativos, exceptuando Linux(en la versión 4.0, sacada al mercado recientemente, sí que se permite poder trabajar sobre esta plataforma), al cual se le puede exportar los trabajos creados pero no así poderlos crear internamente desde su plataforma.

De esta manera el lenguaje de programación que se estudia en este trabajo, es Java, lenguaje de alto nivel orientado a objetos, y creado por *Sun Microsystems* en 1991. Creado en un principio para resolver problemas de comunicación entre diversos aparatos electrónicos, ya que cada aparato llevaba instalado un microprocesador diferente. Este hecho produjo un cambio radical en la investigación, por lo que se escribió un lenguaje de programación nuevo que ayudase a la comunicación de todos estos aparatos sin tener que cambiar sus procesadores, a este nuevo lenguaje se le denominó *Oak*. Pero fue con la explosión de internet, lo que hizo a *Sun* dar un salto con todos sus logros en la investigación de su nuevo lenguaje a la hora de implementarlos en applets, que se ejecutaban en el entorno de Internet. [3]

1.2. Motivación del proyecto.

Cuando a un alumno se le enseña algo por primera vez, o se hace con la ilusión de cuando se aprende a colorear los primeros bocetos de una casa, lo cual habrá niños que no les guste, aunque a la mayoría les gustará, o se hace con la lectura del Quijote a la edad de 12 años, que por experiencia de la gran mayoría, no es recomendable. Todos estos hechos marcan de por vida con que grado de interés será el aprendizaje de algo cuando se ve por primera vez. Así pues, cuando al alumno es iniciado en la programación, el hecho de como se enseña, marcará la reacción que tendrá el alumno en el futuro sobre problemas de índole similar.

La mayoría de los alumnos, ven esta asignatura por primera vez en la universidad, y esto unido a la casi inapreciable base que se tiene de la misma, produce una barrera entre el alumno y los conceptos a estudiar o aprender.

Cuando se ve el primer programa, el sencillo "Hola Mundo", con un par de sentencias de impresión de pantalla o de llamadas a los atributos creados en el mismo; o se analiza lo que se está escribiendo, o se memoriza el tipo de código para luego poder reproducirlo de la misma manera, coma por coma y paréntesis por paréntesis. Es este tipo de actos, en el que aprender de memoria, sin llegar a razonar lo que se tiene que crear, o simplemente revisar para comprender que hace un programa, de donde nace la motivación de hacer este proyecto.

La creación del proyecto viene precedida, por una serie de circunstancias que a lo largo de la carrera se van dando. Dichas circunstancias, no se asimilan hasta que se llega al final de la carrera con algún que otro lastre de más, sabiendo ya tarde, que si la forma de plantearse los problemas de las diversas asignaturas, no sólo de programación, sino de cualquier otro tipo, muy posiblemente el miedo a las asignaturas venideras sería menor que el que se nos infunde solamente viendo el temario de las mismas.

Por lo que habría que hacerse la siguiente pregunta, ¿es necesario la creación de una herramienta que facilite el estudio, y aliente a los alumnos a poner más interés en lo que van a aprender? La respuesta en este caso sería, sí y no. Sí, es necesaria pues con ella se ayuda a los no iniciados en el tema de la programación orientada a objetos, a ver de una forma dinámica y desde el punto de vista de otro alumno, como se ha de razonar en la asignatura y cuales son los principios básicos de la misma. Y no es del todo necesaria, ya que es simplemente un apoyo al cuerpo docente, y cada cual puede tomar esta herramienta como una ayuda o como una mera extensión de lo expuesto durante la jornada lectiva.

Entonces si esto es sólo una herramienta de aprendizaje que tiene que reforzarse con el estudio durante las clases, ¿en qué se debe incidir y que es lo que se puede dejar apartado, para que la herramienta no se convierta en un mero juego para poder pasar el rato y decir que se está estudiando?

La idea de usar como guía del proyecto el libro del autor Dante Alighieri, "La Divina Comedia"[4], es la de usar el modelo de enseñanza que en ella se desarrolla, con Virgilio como guía, el cual va enseñando en cada escena lo más relevante con respecto al pecado en el que están. De esta misma forma, en este trabajo, este mismo guía indica lo que en cada escenario se va a enseñar y da una breve descripción del pecado en el que se va a entrar.

El esquema planteado por Dante Alighieri durante la novela es: Dante, guerrero que busca a su amada, ha de redimirse de todo lo mundano viajando por el infierno, lugar donde se reencontrará con viejos conocidos, personajes ilustres de la historia de la Europa Medieval y habitantes del Averno, que en algunos casos ayudarán a Dante a seguir su viaje, pero que en otros muchos desearán acabar con él por ser de natura no humana al estar allí sin antes haber sido juzgado por San Pedro.

Aunque es el en Purgatorio donde se centra este proyecto, lugar donde el maestro enseña que es lo que le depara cuando se peca, de esta manera Dante ha de aprender de los que allí se hallan, pues para poder ascender un piso más, es necesario limpiarse del pecado pertinente, y de la única forma que se tiene de hacerlo es preguntando, y aquí es donde entra la siguiente pregunta, ¿cuánto de exploración debe de tener un alumno en un tema para que no le resulte del todo monótono o difícil en el aprendizaje?

La motivación que pueda tener un alumno, viene dada por la complejidad de los problemas que al mismo se le plantean, así pues la enseñanza de cualquier materia, ha de ser siempre en grado progresivo desde como actúan los distintos comandos en un mismo programa, hasta que se puede hacer cuando se tiene que escribir un código sólo, en el cual ya participan terceros y se ha de entregar a un superior como solución a un problema mayor.

Para finalizar, comentar que el uso de la herramienta Unity3D, es la que facilita en gran medida la creación de esos escenarios en los cuales cada uno habrá de aprender preguntando a los personajes que en cada escena se encuentran, de tal forma que enseñen a cada usuario como trabaja el lenguaje de programación, desde cuantos tipos primitivos de Java existen, hasta como existen diversos tipos de recorridos de arboles, pasando por listas enlazadas, Nodos de información o sencillamente que es una referencia a *Null*.

1.3. Objetivos que se persiguen.

Los objetivos principales de este proyecto son los siguientes:

1. El objetivo principal del proyecto es facilitar el aprendizaje de una asignatura, relacionada con la programación desarrollada en lenguaje Java, que principalmente no se ha visto anteriormente, y por tanto se intenta ayudar en el primer contacto con la misma.
2. Enseñar desde un punto distinto al que se puede dar durante las clases, para que el alumno pueda ver como se pueden hacer las cosas de varias maneras, ya que la programación tiene varios caminos de pensamiento y por tanto distintas posibilidades de hacer un mismo programa con la misma finalidad.
3. Demostrar las salidas que pueden tener los que en algún futuro se dediquen a la programación profesionalmente, creando videojuegos o aplicaciones para cualquier entorno multiplataforma.
4. Enseñar que con bajo presupuesto se puede hacer un juego educativo, y que con sólo unas nociones básicas de programación, se realizan proyectos básicos con buena presencia con vistas a un público más general.
5. Sentar las bases, para que en otro proyecto, se pueda crear la función multijugador, en la cual los alumnos se podrán conectar mediante una clave de usuario y una contraseña, y puedan explorar en mucha más medida las capacidades que otorga este juego.

1.4. Resumen de la memoria.

Brevemente se expone cuales son los capítulos de este proyecto con una breve descripción de lo que en cada uno de ellos se puede encontrar para facilitar la lectura al usuario:

CAPITULO 1. INTRODUCCIÓN.

- En este apartado se explican las motivaciones del proyecto, se describen brevemente los puntos más representativos del mismo, así como una breve referencia a los programas usados y a su función dentro del trabajo.

CAPITULO 2. ESTADO DEL ARTE.

- Introducción a juegos educativos que tienen como base la programación, convirtiendo estos programas en el primer contacto que se tiene con código fuente, y una breve descripción de los programas más representativos del mercado en el apartado visual de los juegos en 3D. Para finalizar explicando los diversos géneros presentes en el mercado y sus diversas utilidades.

CAPITULO 3. DESARROLLO DEL PROYECTO.

- Capítulo que abarca una introducción más definida sobre la aventura de Dante, junto con las escenas que se han usado y los temas tratados en cada una de ellas.

CAPITULO 4. DISEÑO E IMPLEMENTACIÓN.

- Parte troncal del proyecto en donde se expone el trabajo final, desglosado en distintos apartados de forma que se pueda entender de forma sencilla el trabajo realizado, así como las funciones usadas para conseguir dicho fin.

CAPITULO 5. BATERIA DE PRUEBAS.

- En este capítulo, y a modo de análisis externo de la herramienta, se muestran las opiniones del grupo seleccionado para la primera prueba de la aplicación, para pulir errores de concepto y posibles zonas en las que el juego puede no trabajar correctamente.

CAPITULO 6. TRABAJOS FUTUROS.

- Se proponen nuevas vías de desarrollo para los próximos proyectos que puedan continuar con el trabajo aquí emprendido. Así como una breve muestra de como eran los juegos antes, y como esos mismos juegos han evolucionado a un mundo tridimensional.

PRESUPUESTO.

- Presentación de los presupuestos asociados al proyecto, así como una detallada explicación de los mismos y una breve descripción de cada partida asociada.

ANEXO I. GUÍA BÁSICA DE UNITY3D.

- Introducción básica de como funciona la herramienta que ofrece esta empresa de forma sencillas, para poder empezar a familiarizarse con los controles, las vistas y los paneles de control con los que cuenta.

ANEXO II. GUÍA BÁSICA SOBRE ALGUNAS FUNCIONES DE JAVASCRIPT EN MONODEVELOP.

- Explicación de algunas de las funciones que se usan en el desarrollo de la aventura, y que por ser explícitas del programa Unity, no han sido incluidas en el capítulo cuatro.

CAPÍTULO 2
ESTADO DEL ARTE

La creación de un juego se divide en dos grandes partes, núcleos principales de cualquier juego que se precie, la parte de la programación y la parte visual del juego.

El incipiente incremento del mercado de los juegos en los últimos años ha producido la aparición de múltiples herramientas para poder desarrollarlos de una manera más sencilla, y con el tiempo, poder introducir mejores detalles en los mapas y los escenarios, y lineas arguméntales más complejas y reales.

En los siguientes apartados se hablará de las dos herramientas anteriormente citadas, y un poco de como cada una de ellas influye a su manera en el mercado de los nuevos juegos y su motores de juego.

2.1. Breve introducción a juegos educativos de programación.

Actualmente, se puede encontrar multitud de géneros para juegos sin importar el soporte que lo ejecutará, algunos ejemplos son: los juegos multiplataforma, rol, deportivos, terror - survival o los de tipo educativos. En este apartado se dará una breve introducción a este último género, enfocándolo en todo momento a lo que este proyecto respecta, la educación en la programación.

Según la R.A.E. (Real Academia Española) programar es en términos informáticos, elaborar programas para la resolución de problemas mediante el uso de ordenadores. Basándose en esta idea, el programador antes de ponerse a trabajar, necesita enfocar cual es el objetivo fin de lo que va a programar, valiéndose de toda la información que ha recabado y que le servirá para poder trabajar de una manera más fluida durante el proceso de creación. Pero ¿y si la información con la que trabaja no está correctamente recogida?

Para los problemas que se pueden generar durante la creación de los juegos, muchos organismos saben que es importante poder ayudar desde la base, por lo que se han creado aplicaciones para que los más pequeños de cada casa empiece a “dar sus primeros pasos” en esto de la programación.

La edad para empezar a crear cualquier tipo de programa, no es de relativa importancia, pero si lo es que cuanto antes se empieza en esto, con mayor facilidad se podrá encargar de hacer cada vez trabajos más difíciles y con mayor carga de código.

2.1.1. Alice.

La presentación, en la página web de este programa, es la siguiente: “An educational software that teaches students computer programming in a 3D enviroment”. Se trata pues de una herramienta que enseña a los estudiantes a programar en un entorno dedicado al 3D.

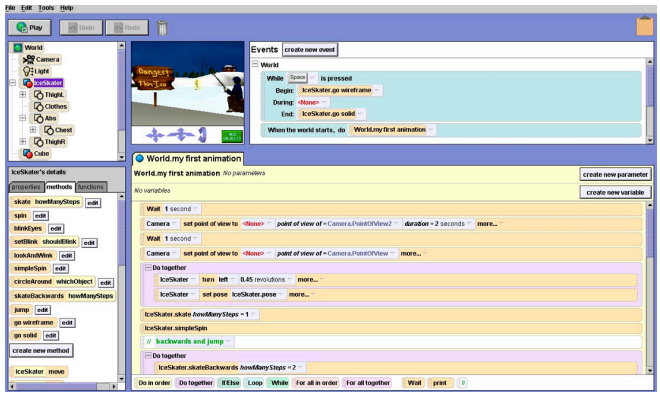


Fig. 2.1. Entorno de trabajo en Alice.

La herramienta es de libre disposición, esto significa que su código fuente es totalmente gratuito para poder trabajar con él, y está diseñada especialmente para ser la primera herramienta a la que un estudiante puede hacer frente en la programación orientada a objetos.

La sencilla interfaz con la que cuenta, permite a los estudiantes poder crear mundos y posteriormente darles vida con la parte programable. Los tipos de lenguaje que soporta este programa son: C++, C# y hasta hace relativamente poco también Java. Lo que hace importante este sistema de aprendizaje, es que según se está programando, se puede ejecutar el programa en tiempo real, y ver si funciona o, es necesario volver a revisarlo para tener una ejecución correcta.

Este tipo de ejercicio facilita en gran medida la labor de los profesores, ya que para los alumnos es más practico ver que hace cada tipo de instrucción y el comportamiento de los objetos que ha creado, y para los docentes, es más fácil enseñar sobre ejemplos prácticos y luego poder trabajar sobre ellos para crear las mejoras, que hacerlo sobre un papel sin poder ejecutarlo y ver si realmente se han equivocado o no.

En EE.UU., es conocido este programa entre aquellos docentes que dan clases sobre programación a los alumnos de primer curso en la iniciación del ensamblador, por esa razón algunas de las más prestigiosas editoriales del gigante americano publican libros sobre la forma en que se debe trabajar y en como se han de ejecutar los programas para conseguir el final deseado.[4]

2.1.2. Snake Wrangling for Kids (Serpiente Wrangling para niños).

“Snake Wrangling for Kids” es un libro electrónico para niños de 8 años en adelante que deseen aprender a programar, haciendo uso del lenguaje Python 3 como base para aprender los conceptos clave.

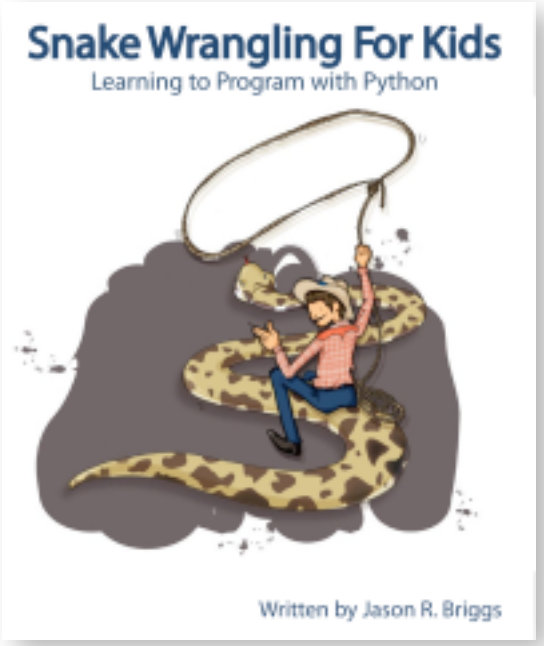


Fig. 2.2. Logo de "Snake Wrangling for Kids".

La versión española está basada en la versión 0.7.7 inglesa, la cual hace uso de Python 3. A partir de ella se han ampliado algunos de los apartados anteriormente implementados, y se han introducido diagramas de flujo para explicar las sentencias alternativas y los bucles. De esta misma manera se han introducido notas a pie de página para explicar el significado en español de las sentencias y las funciones, indicando sus funciones y como han de aplicarse en la ejecución.

Cabe destacar que el lenguaje Python es sencillo y bastante flexible a la hora de realizar programas ejecutables con su código. [6]

2.1.3. Ruby for Kids.

El lenguaje de programación Ruby, comprende un software interactivo con el que se puede aprender a programar en dicho lenguaje, y que su creador Yukihiro Matsumoto, lo define como: "Ruby es un lenguaje simple en apariencia, pero complejo por dentro, como el cuerpo del ser humano". [7]

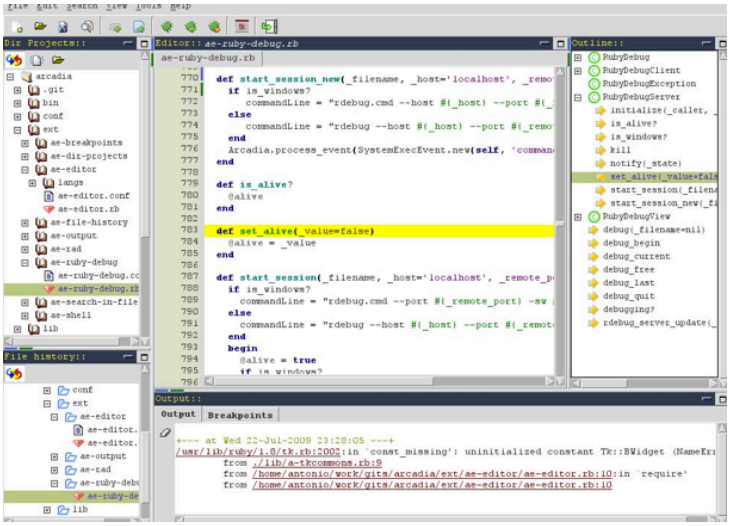


Fig. 2.3.Captura de pantalla del entorno de trabajo.

Antes de la creación de este lenguaje, su creador, decidió que quería crear un código que fuese lo suficientemente poderoso como lo era Perl, y más orientado a objetos como lo es Python.

Con la creación de Ruby, todo es tratado como un objeto. La programación orientada a objetos llama a las propiedades *variables de instancia* y a las acciones *métodos*.

Ruby también es considerado un lenguaje flexible, en donde se permite a los usuarios quitar o agregar funcionalidad a las partes ya existentes y esenciales para su funcionamiento. Pero entrando en un poco más de detalle:

- Es capaz de manejar excepciones como Java o Python, para facilitar su ejecución y estudio.
- Puede cargar bibliotecas de extensión dinámicamente si lo permite el sistema operativo.
- Y una característica muy representativa, es que tiene soporte sobre cualquier tipo de sistema operativo GNU/Linux, Mac OS, Windows, etc., aunque es usado en su mayoría sobre soporte de Linux.

2.1.4. Kodu.

Kodu es un juego creado por Microsoft, y se trata de un programa especialmente creado para la creación de juegos. La principal característica de este software, es que ha sido diseñado principalmente para el manejo de los más pequeños, pero que puede ser usado por cualquier otra persona.

Fig. 2.4. Pantalla de inicio y personaje principal de la aplicación

Lo más representativo de este programa, es la facilidad de interfaz con la que cuenta. El lenguaje está completamente basado en iconos, por lo que es una herramienta muy útil a la hora de poder enseñar a niños.

Los programas están compuestos por páginas, las cuales son descompuestas en reglas, que a si mismo están divididas en acciones y condiciones para los personajes del juego.

Fig. 2.5. Página de ejecución y creación de acciones.

El lenguaje de Kodu, está creado específicamente para el desarrollo de juegos, ya que facilita escenarios pre-programados para su implementación y desarrollo. Los eventos de la ejecución están basados en acciones como por ejemplo, visión, oído y el control del personaje principal que se desee usar.

Como Kodu no se trata de una herramienta de lenguaje profesional, el diseño de su consola de ejecución hace que los primeros pasos en este lenguaje no sean del todo complicados, y que las compilaciones de los algoritmos sean lo más simples posibles.[8]

2.1.5. LEGO MindStorms NXT.

LEGO siempre se ha caracterizado por ser una de las empresas con mayor renombre entre los juegos de construcción, pero en este caso la construcción va un paso más allá. Con este tipo de juego, los jóvenes programadores pueden hacer que sus juguetes cobren vida, con los comandos que ellos mismos hayan introducido en la consola de su creación.

Fig. 2.6. Escorpión y caja de inicio con todos componentes.

La consola es la que da nombre a este particular producto, NXT, con ella se pueden importar los programas creados en el ordenador, y ver cual es su funcionamiento en la figura creada. La conexión es tipo USB o Bluetooth, y es compatible tanto con Windows como con Mac OS, ya que

el soporte de desarrollo lo ofrece LabVIEW, el cual incluye las bases de las instrucciones básicas y algunas guías de como poder crear ciertas acciones partiendo de su código.

El LabVIEW es un intuitivo software de programación gráfica, y es uno de los más usados a la hora de programar reproductores de DVD o MP3, teléfonos móviles o el dato curioso, los airbag de los coches. ¿Quién no querría tener una sonda como la que explora Marte? [9]

2.1.6. Petit Computer.

Normalmente los lenguajes de programación están diseñados para tener su soporte en ordenadores de sobre mesa, ya sean portátiles u ordenadores completos, pero ¿y si se ha de realizar un viaje en donde, durante el trayecto no se puede usar ninguno de ellos y se está a punto de dar con la solución a ese error de ejecución que aparece en la ejecución desde el tercer día?

Para solucionar este pequeño inconveniente, que a día de hoy no suele ser normal, los desarrolladores de DSWare y GAMEBRIDGE han hecho de la consola 3DS una máquina virtual para poder hacer programas con el lenguaje BASIC.

Fig. 2.7. Pantalla de inicio en la 3DSi.

La facilidad de programar en BASIC vuelve a estar presente en la creación de juegos y aplicaciones gracias a este nuevo sistema: El Petit Computer. Viene con una gran muestra de juegos que fueron programados en lenguaje BASIC, para que cada uno pueda ver su implementación y realizar las modificaciones que desee.

La parte más completa de este título, es la posibilidad de exportar el trabajo realizado a un código QR(Quick Response, actualmente el código por antonomasia en Japón y que proporciona una respuesta rápida de procesamiento de la información en ellos contenida), para poderlo compartir con los amigos, o simplemente para exportarlo al ordenador y ver su funcionamiento final. [10]

2.1.7. RoboMind.

De similar temática que el sistema de LEGO Mindstorms, pero solamente en cuanto a forma de trabajar, ya que en este caso, se dispone de un tipo de lenguaje propio, el ROBO, que es de sintaxis mucho más sencilla. [11]

A parte de acercar a las técnicas de programación, también acerca a los usuarios al área de robótica y de la inteligencia artificial. Además es un programa que en sus orígenes sólo estaba disponible en inglés, pero que actualmente también se puede encontrar en español.

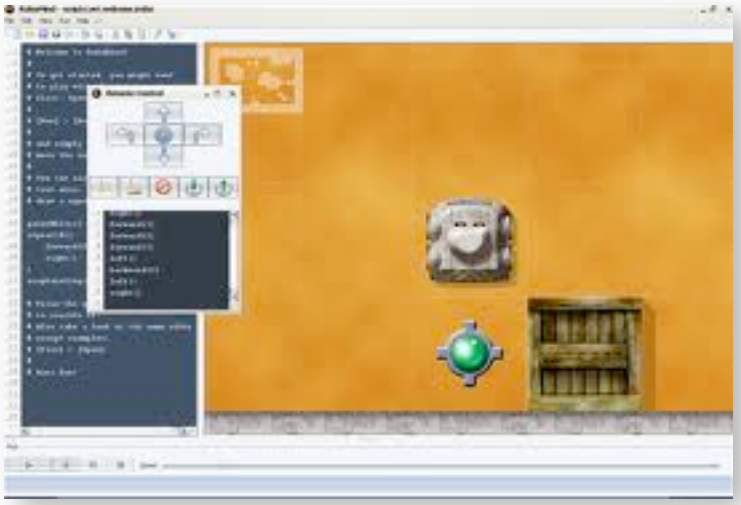


Fig. 2.8. Pantalla principal de programación.

El lenguaje ROBO, está diseñado especialmente para poder empezar lo antes posible a programar y explorar lo realizado lo antes posible, donde todo está enfocado en la programación de los movimientos del robot del juego como principal objetivo. Lo que hace de este programa una herramienta tan llamativa, es que su coste es totalmente gratuito junto con las siguientes características:

- Empezar desde el momento de la instalación. El lenguaje, como se ha indicado previamente, es intuitivo y muy sencillo para poder empezar a implementar los primeros programas, que en un futuro, se convertirán en los primeros pasos del tanque.

- Sin depender de nadie. Los entornos de programación que hacen uso de ROBO, son únicamente de ROBO, no se necesita ningún tipo de compilador externo, ya que es facilitado por el mismo programa.

- Seguro. Lo que se programa, no afecta en ningún momento a ninguna aplicación del ordenador en general.

- Gratis. Como se ha citado anteriormente, el uso de este software, es totalmente gratuito tanto para personas individuales, como para instituciones educativas y su licencia no perece con el tiempo.

- Apto para poder realizar proyectos finales. La herramienta ROBO, está diseñada para poder realizar proyectos interdisciplinares y cursos técnicos sobre la ciencia de la computación.

2.1.8. Scratch.

Este proyecto es un clásico dentro de los académicos y profesionales de la escuela MIT. Desde 2006, es un referente de la programación básica orientada a los más pequeños para empezar a programar con una sencilla herramienta visual, constituida por bloques y animaciones, donde los nuevos programadores pueden ver como va evolucionando su programa.

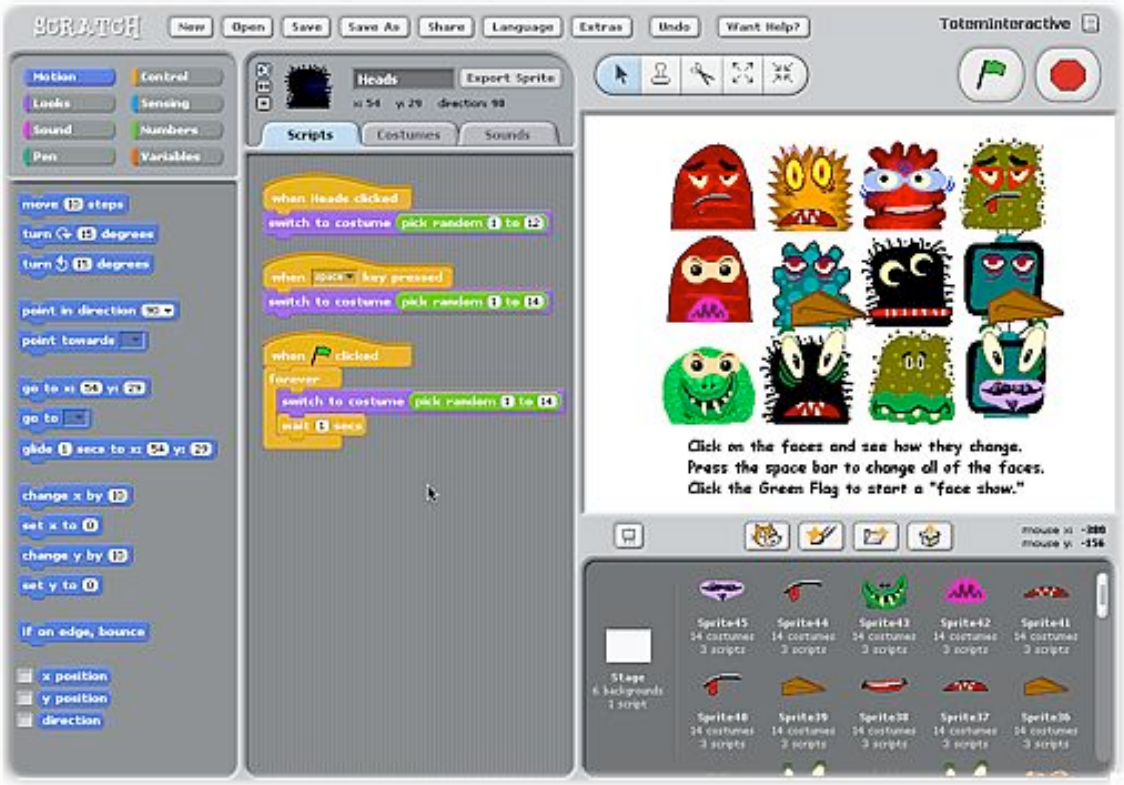


Fig. 2.9. Entorno característico de la casa Scratch.

Este entorno es usado por todo el mundo con diversas aplicaciones, dependiendo de los usos que se les vaya a dar: las escuelas, los museos y los usuarios de todo el mundo. Este sistema está diseñado principalmente para los/las niñas/niños de entre 6 y 16 años, para que empiecen con la programación desde un punto de vista más ameno y sencillo.

El programa como tale es muy sencillo ya que para programar la aplicación, hace uso de símbolos iconográficos, que se denominan “bloques”. La herramienta aprovecha los avances de los diseños de interfaces, como se ha mencionado con anterioridad, para crear una aplicación visualmente agradable y sencilla en los primeros pasos dentro de la programación.

El lenguaje con el que se programa, es Squeak, una implementación libre basada en Smalltalk-80, y es un sistema soportado por cualquier ordenador: Windows, Mac OS X y Linux. [12]

2.1.9. RPG Maker.

En este último caso, se puede observar, como la programación, en muchos de lo casos vistos anteriormente, tiene como finalidad crear un juego o una aplicación de uso compartido para el resto de la comunidad.

Se tiene pues un programa de pago, que permite la creación de multitud de mundos, sistemas de combates y lo más representativo un editor de eventos, similar a los juegos de su género como *Final Fantasy* o *Legend Of Zelda*.

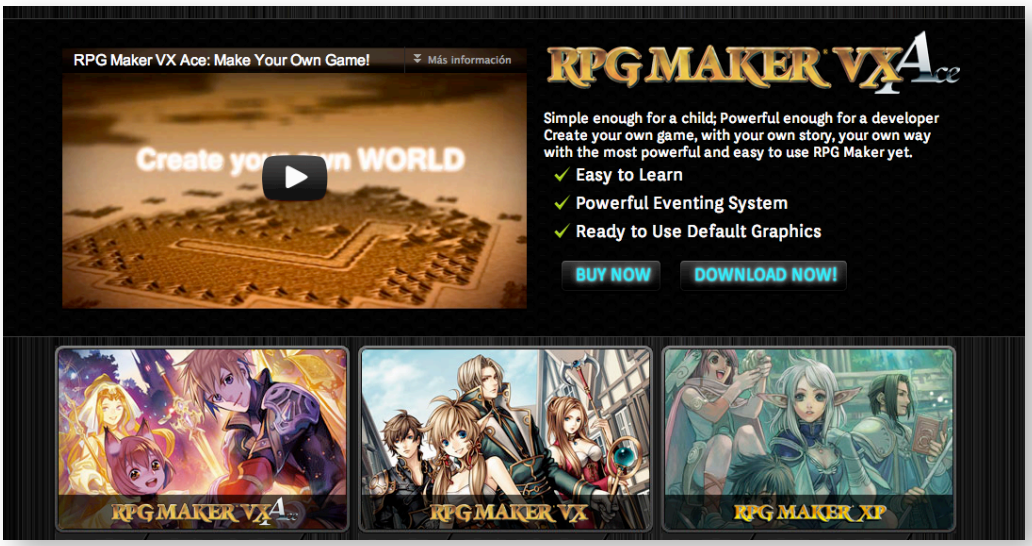


Fig. 2.10. Página principal de RPG Maker.

Lo que convierte este título en uno de los más aclamados en cuanto a la creación de juegos, es la posibilidad de hacer uso de lo que han creado otros, lo que viene de serie en el programa base, o en las últimas versiones, poder crear y añadir tanto nuevos personajes, estructuras visuales o sonidos creados por cada uno.

Es un juego de estilo RPG(Rol Play Game) con visualización en 2D, similar a los primeros Final Fantasy, y está creado en Visual Basic, aunque el lenguaje de programación que utiliza es Ruby.

En este juego, es posible crearse los personajes, hasta un total de 999 de ellos controlados por la máquina, para poder hacer un juego de aventura lo más completo posible. Dentro de cada personaje, se pueden editar las habilidades de cada uno de ellos, según van subiendo de nivel, y posteriormente darle una clase con las características que se desea.

Una vez editado el mapa, y habiendo finalizado la creación de los personajes que se van a poder controlar durante la aventura, se da paso a la programación en Ruby, la cual no es compleja, pues se ofrece un editor de acciones, aunque en las ultimas versiones, también se puede implementar la programación con lenguaje C++, al cual está más habituado la mayoría de los usuarios.

Este título existe desde 1995, pero sus inicios fueron únicamente conocidos en el mundo nipón, y actualmente existen distintas versiones para la creación del amplio catálogo de RPG´s. [13]

2.2. Herramientas para la creación de juegos en 3D.

Una vez se han visto las aplicaciones que existen actualmente en el mercado para la enseñanza de la programación por medio de juegos, es momento de realizar un estudio de la otra parte que depende este proyecto, las herramientas que se usan para la creación de los juegos.

Anteriormente se ha visto como, por ejemplo, con el último programa indicado, el RPG Maker, se podían realizar las dos partes del juego, sin necesidad de otro herramienta externa, que compilase el programa y lo ejecutase externamente.

En esta sección se detallarán algunas herramientas que se usan para la parte visual de los juegos, es decir la animación 3D, y una breve introducción a su funcionamiento.

2.2.1. Ogre 3D.

Ogre vine de las siglas en ingles: Object-Oriented Graphics Rendering Engine, y que corresponden a un motor renderizado de gráficos, orientado a objetos de código abierto.

El programa funciona en la gran mayoría de los sistemas operativos, y el rendimiento de la misma depende de la potencia de cada computadora. Para poder ejecutar lo que se está creando, es necesario un compilador de C++, pero también se ofrece la posibilidad de programar en Python-Ogre y Ogre4j (Java).

Ogre no es un motor de juego en sí mismo, es un motor gráfico que se ha de completar con la introducción de librerías tales como: sonidos, colisiones y entradas de comandos por teclado o ratón.

Fig. 2.11. Torchlight, juego creado totalmente con Ogre 3D.

La comunidad de este motor gráfico es tan amplia, que junto con las API´s proporcionadas por los desarrolladores y los nuevos programas que se crean los usuarios de la aplicación, hacen de esta una herramienta completa y muy sencilla de usar, siempre y cuando se tengan nociones de algunos de los lenguajes necesarios para la programación de los juegos. [14]

2.2.2. Panda 3D.

El siguiente caso, Panda 3D, es un motor de juegos, el cual incluye, gráficos, audio, detección de colisiones y otras herramientas necesarias para la creación de los juegos en 3D.

Fig. 2.12. Instantánea del juego hecho con Panda3D: Ghost Pirate of Vooju Island.

El lenguaje que utiliza esta herramienta es desde sus inicios Python, pero el motor de juego está escrito en C++, por lo que hace que sea necesario un empaquetador externo, para poder exponer la completa funcionalidad del motor en una interfaz que entienda como es Python. Este hecho hace que se conjuguen las potencias de ambos lenguajes, por una parte Python, como desarrollo rápido y la gestión eficiente de la memoria, y por el otro con C++, el cual aporta el uso de un lenguaje compilado en el núcleo del motor de trabajo.

Para concluir, indicar que está herramienta es una creación de la casa Disney, y fue la encargada de dar vida a las atracciones de realidad virtual de los parques que en ellos se encuentran. Posteriormente, y dado el renombre que tuvo en sus atracciones, decide liberar su código, para de alguna manera, poder contribuir al desarrollo de más aplicaciones y estudios desde los distintos órganos públicos.

2.2.3. JMonkeyEngine.

La abreviatura de este editor de juegos es más conocido que su nombre, JME, y como los anteriormente citados, también se trata de un motor de creación de juegos de carácter libre, pero en este caso el lenguaje con el que se programa es Java.

Posteriormente, fue usado cada vez por más programadores, que poco a poco ampliaban sus funciones, y se convirtió en una herramienta con modernos efectos gráficos, para finalmente ser una plataforma estable, dentro del mercado de la creación de juegos.

Fig. 2.13. Trabajo realizado en el editor JMonkeyEngine.

La arquitectura que hace uso, es de tipo árbol, lo que facilita el uso de los nodos con independencia de lo que está pasando en otro nodo. Así de esta manera, un padre puede tener infinitos nodos, pero un hijo solamente tendrá un padre. Este tipo de proceso, permite que la escena realmente importante, esté en primer plano, mientras que lo que no es necesario, o está ocupando memoria constantemente, no lo esté procesando constantemente, lo que agiliza en gran medida el procesamiento de la información sin llegar a saturar el programa en sí.

Dentro de todos los programas que se están viendo, es el que menos aceptación tiene entre los usuarios más conservadores, ya que la facilidad de creación de juegos, no es de las más sencillas, pudiendo elegir entre cualquier otro de los motores explicados con anterioridad.

2.2.4. UDK.

¿Quién no conoce el aclamado juego en primera persona Unreal Tournament? Si no se conoce, indicar que es y sigue siendo uno de los juegos en linea con más seguidores de la historia y que cada cierto tiempo, y con la mejora de sus motores gráficos, sacan versiones mejoradas, lo que provoca la reactivación de este título en el mercado.

Pero volviendo al tema que aquí se está tratando, el UDK (Unreal Development Kit), es una herramienta para la creación de juegos, a alto nivel de trabajo, es decir, con grandes producciones que posteriormente verán la luz como juegos comerciales.

Fig. 2.14. Renderizado del templo mediante UDK.

El programa en si ofrece multitud de aplicaciones y formas de trabajar para delicia de los más emprendedores. Se puede editar, crear animaciones, juegos en red, y multitud de elementos asociados a la animación y a los videojuegos.

Dentro de esta herramienta se puede encontrar un sistema propio para la creación de animaciones, junto con un lenguaje de programación único para ejecutar los scripts, el UnrealScript, y un sistema de ayuda con los nuevos scripts, el Kismet, para aquellos que no tengan conocimientos de programación.

Indicar que el equipo de Unreal Development, es actualmente una de la mayores empresas que dan soporte a la industria de los videojuegos, ya que con su sencillo motor de trabajo, hace posible la creación de juegos nuevos, desarrollados por las empresas como son el ejemplo de: *Propaganda Games*, con su aclamado juego en primera persona *Turok*; el colosal *BioShock* con su argumento innovador y sus gráficos, desarrollado por *Irrational Games* o el juego del año en 2012 *Batman: Arkham Asylum* de la compañía Rocksteady Studios entre otros.

Como aporte final al estudio de algunas de las herramientas más populares usadas para la creación de juegos en 3D, cabe resaltar que existen multitud de ellas, y que sólo se han puesto las más representativas en el entorno de la creación de los juegos, y que todo depende del lenguaje de programación con el que se vaya a trabajar y del aspecto visual que más agrade al programador, pues como se ha comentado, cada uno de ellos lo hace de una manera distinta, pero todos con la misma finalidad, crear un juego para deleite de otros usuarios.

2.3. Géneros de los juegos.

En la actualidad existen multitud de géneros dentro de la categoría de los juegos, por ejemplo:

- Juegos de plataformas, contribuyen a la orientación espacial, los más conocidos en el mercado son *Super Mario Bros*. de la compañía *Nintendo* y *Sonic: the hedgehog* de *Sega*.

- Juegos de puzzles, en este caso se desarrollan la percepción y la atención que ha de tener el usuario para poder superar un enigma concreto, uno de los más comunes en este género es el famoso *Tetris*, que aún a día de hoy tiene millones de aficionados en todo el mundo.

- Juegos de tipo simulador, ya sean de estilo deportivo o de conducción, permiten experimentar e investigar como funcionan las máquinas o como reaccionar ante ciertas situaciones imprevistas, como pueden ser *Wii Sports* o *Flight Simulator*, respectivamente;.

- Juegos de estrategia, en la mayoría exigen administrar recursos escasos, como puedan ser alimentos, armamento, dinero o fuentes de poder y se intenta prever lo que puede hacer el adversario y actuar en consecuencia, uno de los títulos más famosos de este género es el *Age of Empires*, donde se puede observar la evolución del ser humano desde la Edad de Piedra, hasta la Edad de Hierro.

- Finalmente los juegos de rol/aventuras y educativos, que son los que aquí se van a desarrollar detalladamente.

Realmente ¿qué se entiende por un juego de rol/aventuras? Los juegos de esta índole, se caracterizan por ser de tipo investigación, exploración, resolución de rompecabezas, interacción con personajes de la aventura y una visión más centrada en la historia que se da, y no tanto en los reflejos que pueda poseer el usuario.

La verdadera diferencia que existe entre este género de juegos y el resto, se trata del enfoque que se da a la historia, pudiendo ser de terror, misterio, acción o ciencia afición. Algunos de los juegos más destacados en este género son: *Myst*, *The Legend of Zelda* e *Indiana Jones and the Fate of Atlantis* entre otros.

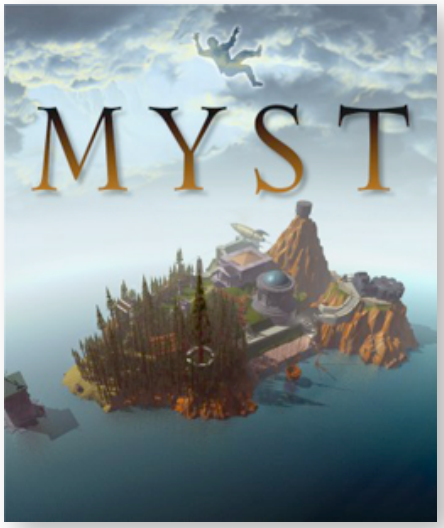


Fig. 2.15. Portada del juego aventura gráfica Myst.

También es posible encontrar en el mercado mezclas de este tipo de género con los de acción, ya que es muy fácil pasar de un tipo lineal de juego a uno con mayor trasfondo sobre la historia, en este caso se puede encontrar *Batman: Arkham City*, videojuego de la compañía *Rocksteady Studios* y sacado al mercado en 2011.

Fig. 2.16. Ilustración del aclamado juego del año 2011: *Batman:Arkam City*.

Y por otra parte se encuentran los juegos educativos, y como se ha preguntado con anterioridad ¿cómo se puede reconocer un juego educativo en la actualidad?

El apartado educativo era un completo desconocido para los jugadores más experimentados en la materia, pero que se está haciendo cada vez más hueco entre los nuevos jugadores y algunos más veteranos.

Las grandes compañías sabe que han de llegar cada vez más a un público específico de la comunidad, y por esa misma razón han escuchado las voces de usuarios que pedían más acceso a los más pequeños de la casa. La mayoría de estas aventuras transcurren como en los géneros de aventuras, pero en este caso el trasfondo de la historia es el poder comprender lo que se pide y lo que se está intentando enseñar.

Inicialmente, fue la compañía nipona Nintendo la que descubrió un filón en este tipo de juegos, para poder exprimir al máximo el rendimiento de su nueva consola, la Wii, ya que con ella se podrían hacer actividades en familia, y con la portátil de la casa, la Nintendo Ds, consiguió llegar a un público que no se encontraba entre el sector de los grandes consumidores de videojuego. Entre los juegos destacados dentro de esta categoría, se pueden encontrar: *Naraba*, *Brain Training*, *Where in the world is Carmen Sandiego, Wii Sports* o *The typing of Dead 2*, en este último título, los "salvadores", en vez de llevar pistolas como sucede en el juego del que deriva, usan el teclado como arma contra las hordas de zombies.

Fig. 2.17 Escena del juego *The typing of Dead 2*.

El mercado de los juegos está inundado de títulos en los cuales la violencia, ya sea del tipo físico o verbal, es cada día más común, de esta manera, los pocos juegos que existen educativos en su descripción de la parte trasera se puede observar "*la alternativa inteligente a los juegos violentos*". [18]

Pero, ¿cuánto de educativo puede resultar un videojuego, si normalmente se comenta que sólo sirven para pasar el tiempo libre? Dependiendo del tipo de juego con el que se trabaje, se pueden analizar los distintos beneficios educativos que ofrecen.

Los juegos de aventuras, como se ha indicado en los párrafos anteriores, pueden ser de cualquier estilo, desde terror, hasta acción, siempre y cuando no pase a ser un juego en el que la acción sobre cargue demasiado el juego y pierda su esencia. Por lo que si un usuario se pone a jugar con el *Brain Training*, lo que está mejorando es la capacidad de memoria, resolución de problemas y reflejos oculares.

La gran mayoría de los juegos educativos actuales están dirigidos al público más pequeño, ya que en esas edades son como esponjas, y aprenden de todo lo que les rodea, por lo que la mejor manera de acercarse a dicho público es con la creación de juegos o aplicaciones, ya que la mayoría de los pequeños hacen más uso de las tabletas digitales y los teléfonos móviles que poseen sus progenitores.

Fig. 2.18. Tableta de uso exclusivo para niños de TVE.

El uso educativo que ofrecen los juegos de este estilo son:

- **Resolución de problemas mediante le aprendizaje:** a la vez que se atrae al usuario a un entorno conocido, también se le proponen retos para que no se convierta en una aventura lineal de sólo pulsar botones.

Fig. 2.19. Puzzle del *Skyrim* que se tiene que resolver con ayuda de las herramientas ofrecidas.

- **Ayuda en el propio juego para poder resolver las dudas de como resolver un puzzle:** durante el juego, es necesario en muchos casos recurrir a terceros para poder resolver un reto, por lo que esto implica pedir ayuda, que en la realidad es recurrir a otros usuarios mediante la comunicación.

Fig. 2.20. Juego DiscWorld, en donde la resolución de puzzles preguntando es esencial.

- **Educación social:** el objetivo en mucho casos, también recae en la educación del entorno que nos rodea, con respecto a las señales de tráfico, las drogas o los entornos familiares hostiles, y como se ha de actuar en los casos que se exponen.

Fig. 2.21. Juego que enseña las señales de tráfico y como se han de respetar.

2.4. Estado actual de los juegos educativos.

La industria de los videojuegos educativos en la actualidad, se podría afirmar que está empezando a nacer. Todos los juegos en mayor o menor medida pueden llegar a ser útiles en cuanto a pautas de aprendizaje, ya que además de enseñar, potencian distintas habilidades tanto mentales como de psicomotricidad.

El género de juego, actualmente está centrado en lo que a público infantil respecta, ya que es en ese momento de la vida, cuando más información se asimila, y se empieza a tener contacto con las primeras consolas o programas educativos instalados en las propias aulas de las escuelas, o en los dispositivos móviles como ya se vio en el apartado anterior.

En cuanto a los distintos títulos educativos que podemos encontrar en el mercado, o que hayan marcado un antes y un después en la era de los juegos podemos encontrar:

- ***Pipo*(el videojuego)**. Se trata de un juego que recorre la gran mayoría de las temáticas que se estudian en las aulas: Geografía, Matemáticas, etc. El protagonista ha de hablar y avanzar sobre los problemas que se le exponen a lo largo del juego. En este título, lo que se desarrolla, es la memoria del usuario, ya que tiene que recordar quien ha pedido cierta información, y según que favores, se pueden activar otros enigmas. [19]

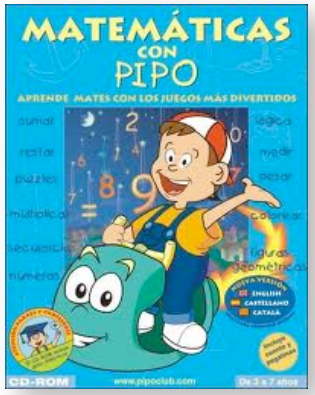


Fig. 2.22. El título más famoso para poder aprender divirtiéndose.

- ***Brain Training del Dr. Kawashima ¿Cuántos años tiene tu cerebro?*** En este título, revolucionario en el mercado de los juegos educativos, inicialmente se presentan sencillos ejercicios, desbloqueándose nuevos una vez se han superado los anteriores, aumentando con ello la complejidad de los problemas. A este cartucho, se le asocia la capacidad de poder realizar cálculos, retener datos o realizar actividades de lectura y ortografía. [20]

Fig. 2.23. El Dr. Kawashima anima a practicar todos los días con su juego.

- ***Where in the world is Carmen Sandiego***. Juego basado en la mítica serie de 1983, en la que se realizaban viajes por todo el mundo. Aquí la protagonista es perseguida por dos jóvenes, mientras se explican monumentos y datos de cultura general sobre los países que se visitan. Haciendo uso de un almanaque mundial y mapas de los países, los jugadores debían resolver preguntas de tipo geográficas.[21]

Fig. 2.24. Serie de tv y posterior juego que recorría el mundo enseñando distintas ubicaciones.

- ***The typing of Dead 2***. Ambientado en el shooter de temática zombie, *The House of the Dead*, en donde el objetivo es matar bestias a cañonazos, en el título educativo, se sustituye la escopeta por un teclado de ordenador, en donde se ha teclear lo más deprisa posible las letras para acabar con ellos. Se intenta dar rapidez de lectura, de asimilación, y de concentración, ya que pulsar la tecla incorrecta reanuda la secuencia de escritura.

- ***Math Blaster***. Temática espacial, en donde las matemáticas controla al personaje y sus acciones. Se subdivide en distintos niveles de matemáticas, desde las más simples operaciones de suma y resta, hasta ecuaciones más complejas para poder avanzar en esta aventura arcade.[22]

Fig. 2.25. Resolución de problemas matemáticos para acabar con los Aliens.

Pero no sólo en juegos propiamente educativos se pueden encontrar pautas de aprendizaje o de memoria. Los juego comerciales distribuidos por las grandes empresas, también se pueden considerar de tipo educativo, o más bien de tipo de aprendizaje por memoria o ensayo error:

- ***Wii Sport Resort***. Juego donde se prima la habilidad del jugador, y el carácter deportivo, ya que potencia la practica de actividades en equipo y las habilidades sociales. [23]

- ***El Profesor Layton***. En cualquiera de sus entregas, en España únicamente se han publicado 4 juegos, pero en Japón es todo un fenómeno de masas. En él se han de resolver acertijos, que servirán para poder continuar con la trama principal del juego. Fomenta la

concentración para hacer frente a problemas, o puzzles como son más conocidos, mientras se avanza.[24]

Fig. 2.26. ¿Cuál será el valor necesario?

- ***PlayEnglish***. Título de acción, se ha de resolver un misterio por todo el viejo continente. En él se pone a prueba el conocimiento de la lengua inglesa a la vez que se le enseñan nuevas palabras y los distintos acentos existentes en una misma lengua, ya que es imprescindible hablar con distintos personajes, cada uno de ellos proveniente de distintas partes del mundo. [25]

- ***Pokémon***. Conocido juego de captura de criaturas en un mundo virtual. En él prima hacerse con todos los "pokémon's" que se pueda, existiendo distintos tipos de especies, lo que obliga al usuario a saberse sus características y a formar una estrategia para poder enfrentarse a enemigos propuestos y seguir con el hilo argumental.[26]

Fig. 2.27. Fuego contra volador ¿quién ganará?

- ***Sid Meier's Civilitation***. Juego de temática histórica, basado en turnos. El jugador debe aprender rápido, saber que tipo de gobierno le favorece más a su estilo de juego, los monumentos que ha de crear para poder avanzar en la historia, y no siempre se ha de limitar a realizar lo que realmente pasase en la historia. [27]

En este apartado sólo se han mostrado algunos de los juegos que pueden ser considerados de cualquier otro género, pero que en el fondo siempre representan una base educacional, lo que implica que el usuario se desarrollará mientras siga jugando al juego.

2.5. Utilidad de los juegos.

Cuando se habla de videojuegos, nos imaginamos a usuarios con gafas, camisetas con mensajes que sólo unos pocos pueden entender, o similares, pero ¿quién se puede imaginar a un cirujano profesional jugando 1 hora antes de una operación al *Street Fighter*?, pues sí, y todo gracias a la eficiencia que llegan a ofrecer los videojuegos.

"Los videojuegos aumentan el potencial de resolución de los problemas", con estas palabras se referían varios investigadores, reunidos en la **Convención de la Asociación Americana Psicológica**, celebrada en Boston (Massachusetts) en 2008.

Uno de los trabajos, desarrollado en la Universidad de Frodham, se centró en 122 estudiantes, a los cuales se les pidió pensar en voz alta durante 20 minutos a la vez que estaban estrenando un juego por primera vez. Los expertos dedujeron, que este tipo de actividades mejoraba las habilidades cognoscitivas y la percepción.

Otro de los juegos en los que se realizó el estudio, es el aclamado "*World of Warcraft*", con el cual los científicos determinaron que "puede mejorar el pensamiento científico", se pone entre comillas, ya que sin unas buenas bases y un aprendizaje sobre la materia, de forma más específica, se quedaría en eso, el "puede".

En la misma convención citada anteriormente, y basándose en el programa realizado con los estudiantes y el juego, Douglas Gentile, profesor de psicología de la Universidad de Iowa, presentó un estudio sobre el impacto de los videojuegos en la instrucción de los cirujanos en el siglo XXI.

En el estudio, participaron 33 cirujanos(15 hombres y 18 mujeres), del Centro Médico Beth Israel. La investigación se centro en cuatro juegos uno de ellos especialmente diseñado para ellos y usado por miles de cirujanos en todo el mundo, ***Top Gun*** [28], un programa de destreza en laparoscopia, y otros tres muy distintos a los que no estaban acostumbrados a usar: *Super Money Ball 2*, *Star Wars Racer Revenge* y *Silent Scope*, cada uno con un desarrollo de las destrezas distintas para lograr un objetivo o meta final.

Los cirujanos que participaron, durante un mínimo de tres horas antes de las operaciones, fueron un 27% más rápido en la cirugía y sus errores se vieron reducidos a un 37%. En resumen, los estudios indicaron que los cirujanos habían mejorado los resultados en un 42%, con respecto a los que no habían practicado en los videojuegos. [29]

2.6. Conclusiones.

La creación de un juego es la manera más fácil, que no sencilla, de acercarse al público adolescente y jugador, en la actualidad. No es tarea sencilla, ya que para poder realizar un juego de mínimos aceptables, es necesario un gran equipo y mucho tiempo para el desarrollo de la aplicación.

El uso de la herramienta Unity3D, ha sido el preludio para querer aprender más sobre este mundo que tantas oportunidades da, y que se puede convertir en un trabajo en el futuro.

La influencia más relevante ha sido la aportada por el juego de *Batman*, en cualquiera de sus dos versiones, el *Arkam City* o el *Arkam Asylum*. En cualquiera de estos juegos se pueden observar varias características claves para el desarrollo de una aventura con tintes educativos: una de ellas es

el tener que preguntar a la gente para poder avanzar en la historia, y la otra, según se avanza, se aprenden cosas independientemente del tema que se esté tratando, en conclusión un aprendizaje en paralelo.

El tema de la programación es del todo delicado, si no se explica correctamente, es muy difícil hacer frente a los programas que tengan una mayor complejidad cuya escala sea ascendente. La primera vez que se ve un lenguaje, ha de ser de una forma abierta y sin prejuicios de ningún tipo, por eso la idea de crear un juego desde el punto de vista del alumno es tan importante, pues se enfatizan los puntos claves de dicha estructura y como se pueden abordar los problemas que se plantean desde el otro lado de la mesa.

Un juego que realmente es educativo, es desde el punto de vista educacional, aquel que cuando se deja de usar, aun el usuario es capaz de recordar lo que en él se ha aprendido o se ha realizado, por lo que el juego intenta poner personajes característicos del mundo en general, para una vez desconectado del juego se recuerde en la mayor medida posible lo que les ha indicado, que no enseñado, como en el juego de *Pokémon,* en el que cualquier chaval que ha jugado se sabe de memoria cuales son los ataques de sus criaturas, así como su tipo y su debilidad.

Con todo lo explicado anteriormente, se pretende concienciar a los docentes, que actualmente las clases no sólo se dan dentro del aula, sino que se puede seguir dando desde cualquier sitio, con aplicaciones sobre las materias y juegos que ayuden a profundizar más en materias que posteriormente son necesarias para hacer frente a los trabajos que se expongan.

Capítulo 3

Desarrollo del proyecto

3.1. ¿Por dónde se empieza un juego?

A la hora de poder crear un juego, esa es la pregunta principal, junto con posiblemente: ¿qué es lo que más vende actualmente?, o ¿qué ha tener un juego para que triunfe: originalidad o ser parte de algo que ya existe(como por ejemplo las aventuras de algún personaje de cómic)? Para la creación de un juego basado en las conversaciones de tipo educativo, que es lo que aquí se tiene, se pueden coger dos caminos:

I. Dar vida a una historia desde sus comienzos, es decir, que sea una historia del todo creativa y que no se haya visto nada igual con anterioridad. El problema de este tipo de historias, es que se han de pensar junto con una gran cantidad de mentes creativas, para poder trazar un hilo argumental lo bastante sólido como para poder enganchar a los usuarios del juego y que la actividad en ellos no decaiga. En este caso, por ejemplo se pueden ver los juegos de *Monkey Island, Runaways*, o sin ser una aventura gráfica, y más bien un RPG, la exitosamente saga de *Pokémon*, en la que destaca la cantidad de historias cada vez más innovadoras, pues ningún juego, aunque se trate siempre de una evolución de los ya editados, tiene que ver en la historia con el resto, por tratarse siempre de lugares totalmente nuevos.

II. Basarse en una historia ya creada, y perfeccionarla o darle un giro de los acontecimientos para hacerla más atractiva al usuario. Este tipo de inventiva, no necesita un alto grado de imaginación, a no ser que el juego trate de un libro o de una película, lo cual lo único que se ha de pensar es el tipo de argumento que se tiene e intentar adaptarlo al juego de la forma más sencilla posible. De esta forma se pueden incluir en los propios juegos, escenas eliminadas o partes jugables distintas a las vistas, para poder hacer más atractivo el juego al consumidor y tener un alto porcentaje de ventas.

Actualmente se sabe que la producción a gran escala de un juego que represente una película, sólo para aprovechar el tirón de esta, no es sino una practica muy extendida ahora y que sólo es usada con finalidad el público más joven.

Este tipo de juegos tienden a lo monótono y a la poca imaginación que en ellos se plasman, debido a que deben ser fieles reproducciones a lo que se refieren. Por lo que se puede constatar, cuanto mayor sea el nivel de innovación dentro de una historia, aunque esté basada en una anterior, su éxito puede verse o bien truncado, pues no aporta ningún tipo de avance a lo que ya existía, o al contrario, el avance con la anterior historia es tal que se pueden realizar cosas que antes no se podían, en este caso tenemos el ya citado juego "*Batman: Arkham Asylum*", que supuso un avance en los juegos de la época, y que con su segunda parte "*Batman: Arkham City*", ha conseguido proclamarse el mejor juego del año, ya que innova tanto en estética, jugabilidad y los aspectos visuales presentados al usuario.

También se puede observar como otros juegos que tienen sus inicios en la gran pantalla, no suponen un fracaso colectivo, gracias a la colaboración de grandes guionistas. Tal es el ejemplo de "*Indiana Jones and the fate of the Atlantis*", que se basa en el personaje y la ambientación de las películas, para crear totalmente una nueva aventura, la cual dio lugar a incluirla dentro de las grandes aventuras gráficas de culto.

3.2. Historia de Dante.

En esta aventura, se asientan las bases para la creación de una aventura educativa haciendo uso de la herramienta Unity, y que tiene por propósito repasar, y ayudar con las bases de la programación orientada a objetos de Java, la cual equivale a la asignatura de Programación de primero, generalmente común a todas las carreras relacionadas con la ingeniería de Informática y de Telecomunicaciones. Como se puede observar, este tipo de asignatura es del tipo continuo, es decir, se han de tener unos buenos conocimientos desde la base, o no se podrán comprender con facilidad los temas que se traten más adelante.

Se ha de pensar por tanto un lugar en donde se pueda recrear un ambiente similar al necesario en las aulas, con un tutor que guíe a los alumnos, y unos alumnos que necesitan ser guiados por el que imparte la materia. Se puede observar que en un capítulo entero de "**La Divina Comedia**", de Dante Alighieri, específicamente la parte en la que se encuentran en el purgatorio, se dan las condiciones necesarias para poder empezar con la historia del juego.

El purgatorio de Dante, tiene la forma de un cono, el cual se encuentra dividido en: 7 círculos, cada uno de ellos representa un pecado capital, y 2 menos significativos, que son la entrada al purgatorio, regentada por Caronte, y el paraíso terrenal, lugar en donde se encuentra con su amada Beatriz, donde se da por finalizado el libro del purgatorio, y empieza el del paraíso.

En cada uno de los distintos pisos que componen el purgatorio, se imparten castigos a los que en vida pecaron y han de redimirse con sus lamentos y sacrifico.

Fig. 3.1. Cono que representa los pecados y sus castigos en italiano.

Puertas del purgatorio: en ellas se encuentra Catón, un pagano que ha sido nombrado por Dios como el guardián al pie de la montaña para no dejar subir a nadie que no lo merezca. En este lugar se encuentran los tardíos arrepentidos, y los excomulgados de la fe cristiana.

En el primer piso, se encuentran los **SOBERBIOS**, en donde se pueden observar cuerpos doblados por el gran peso de las piedras que cargan sobre su espalda.

En la segunda planta, se puede observar a los **ENVIDIOSOS**, los cuales para no poder ver los bienes del prójimo, tienen los ojos cosidos, como los halcones en el arte de la cetrería haciendo de esta manera que sólo dependiesen de su sentido auditivo.

Los **IRACUNDOS**, habitan la tercera terraza de este fantasmagórico mundo, y sus habitantes están rodeados de humaredas de acre, que simboliza la ceguera que puede llegar a producir la ira sin pensar racionalmente.

La cuarta planta está llena de **PEREZOSOS**, sus almas están condenadas a correr sin ninguna meta definida, y no pueden pararse para poder responder a las preguntas del viajero Dante.

En la quinta grada de este espacio, descansan los **AVARICIOSOS**, están condenados a estar postrados en posición boca abajo sin posibilidad de levantar la cabeza.

La purificación de los **GOLOSOS**, se produce en el sexto estadio del purgatorio, y todos los que allí se encuentran, ávidos de no poder parar de comer, se les condena a ver árboles llenos de frutos jugosos. En este punto se plantea la duda de la separación cristiana de la naturaleza del alma y su relación al cuerpo vivo.

El la penúltima terraza, esta el peor de los pecados, los **LUJURIOSOS**, almas arrepentidas del deseo sexual, que están ante un muro de fuego, el cual separa el mundo el purgatorio del paraíso terrenal, y por el que se ha de pasar, sólo y sólo sí, se está totalmente limpio de todo pecado.

Por último llegan al Paraíso Terrenal, o Jardín del Edén, del cual fueron expulsados Adán y Eva, y donde es recibido por una comitiva dirigida por Beatriz, que le hace beber del río Lethe, cuyo poder borra la memoria de los pecados anteriores, y del río Eunoë, que le devuelve la memoria y le prepara para el ascenso junto a su amada hacia el Paraíso.

Puesto que la aventura con la que se ha deseado trabajar es esta parte y no otra como el Infierno(tema tratado en este completo trabajo [30]), se debe concretar que tipo de tema se acerca más a la terraza en la que se encuentra, aunque sea de forma sencilla. El orden que se ha usado no es el mismo que se ha indicado con anterioridad, pues como se acaba de citar, no es tan sencillo meter a los lujuriosos con el tema inicial de la programación, por ejemplo.

La idea inicial es que el usuario ha de hablar con todos los personajes del mapa, para poder aprender todos los conceptos, siendo temas básicos en comparación con lo que se trabaja en clase, y una vez hablado con todos los personajes, responder a una pregunta hecha por el ángel que custodia cada terraza para poder avanzar en la aventura.

La facilidad con la que se ha propuesto el juego, no ofrece “casi” ninguna posibilidad de quedarse atascado en la trama del juego (el casi ha de ir entre comillas, ya que incluso en los juegos de mayor renombre, se supone un 1% de posibilidades de que el usuario se quede atascado o que el juego en sí mismo crea algún error imprevisto y sea necesario reiniciar de una forma inesperada).

La guía de Virgilio durante el juego es la misma que se ofrece en la realidad por los tutores de la asignatura, indicando en todo momento cuando se cometen errores y como se han de abordar, y aunque el personaje de Estacio, les acompaña durante una parte del viaje, su presencia en el libro es significativa, únicamente como apoyo para Virgilio en la travesía de viaje al Paraíso.

El paso de un mapa a otro se ha de realizar mediante la respuesta que cada usuario dará a la pregunta que le postule el ángel pertinente de cada terraza. Las preguntas son una batería de pruebas

en las cales a cada una de ellas está asociada una única respuesta. El juego terminará con la última pregunta, del último mapa, del último ángel.

3.3. Planificación del proyecto.

La finalidad de este proyecto en concreto, tiene como fin crear una aventura gráfica de primera persona, haciendo uso para ello de Javascript y Unity 3D. Por lo que las tareas que se han de desarrollar bajo un cierto grado de atención importante son:

✓ Diseño de la aventura.

✓ Ser capaz de desarrollar escenas que sean lo más atractivas para los usuarios, en las cuales se puedan observar a primera vista el pecado en el que se encuentran y que sea sencillo ubicar a los personajes para interactuar más cómodamente.

✓ En un principio se optó por personajes en forma esférica, hexagonal, o piramidal, debido al poco tiempo del que se disponía para poder realizar la aplicación, pero se encontró una página [31], en la cual de forma gratuita se pudieron descargar personajes más creíbles.

✓ La implementación se decidió realizarla sobre el soporte de Javascript, con la posibilidad de trabajar con el complemento de XML, pero debido a un estudio más exhaustivo de como se podría realizar de una forma cómoda, para una posterior traducción a otros idiomas, se acabó trabajando con un lenguaje más conocido y sencillo de utilizar, JavaScript, ya que se trataba de un lenguaje similar al que se había usado durante el curso y era más factible para una posible ampliación del juego más adelante.

✓ Diseño de todo el guión, y los diálogos, así como la ambientación de cada mapa acorde al pecado en el que se encontraba.

✓ Para poder realizar una obra más completa, anteriormente se presenta una guía de lo que se ha de conocer para poder comprender mejor la asignatura, a modo de base para los que desean ver un punto de vista distinto sobre los mismos contenidos[32], la cual ha sido usada para la realización de la parte de Java que se ha de conocer.

✓ En un principio se decidió incluir monumentos descargados de Google 3D, pero el peso que suponían para cada escena, apartó la idea, optando por los objetos deseados de una manera mas brusca.

✓ Con referencia a los diálogos de cada personaje para poder mantener la atención del usuario en todo momento, se optó por asociar a cada personaje una tecla(siempre se indica al inicio de la conversación).

✓ Las respuestas del ángel son de tipo múltiple, esto indica que la respuesta estará contenida entre un cierto número de soluciones propuestas, en el cual si se responde mal, se obliga al usuario a repetir el mapa en el que se encuentra, de tal manera que al volver al propio ángel, la respuesta puede ser la misma o haber cambiado por otra que se habría introducido anteriormente en la batería de preguntas y respuestas.

✓ Incluir sonidos ambientales fue una decisión por parte del equipo de desarrollo, por la cual se pretende que el juego no se convierta en un desarrollo simple de las escenas. Un

problema que no se ha solventado, fue el relacionado con la inclusión de voces de personajes, ya que no se puede hacer que hablen una vez, y si es necesario volver a hablar con ellos, sólo indicar lo que realmente era importante.

Una vez conocidos los aspectos principales y como se han de afrontar desde cero, la metodología que se ha usado es la conocida como espiral: se determinan los objetivos que se desean cumplir y se estudia cuales son las alternativas más exitosas, una vez concluido ambos procesos, se estudia el riesgo que dicha implementación puede tener en la finalidad del trabajo para posteriormente planificar lo que realmente se va a introducir, y para concluir, se desarrolla el proyecto, se ejecuta y se comprueba que cumple los requisitos, si estos son afirmativos, se valida, y si por el contrario no son del todo aceptables, se ha de volver al primer paso y empezar a ver en que ha fallado.

Conocer que se está haciendo al respecto en estos momentos, cuales pueden ser las herramientas que facilitan todo el trabajo que se quiere hacer, como se ha realizar para poder llegar a un público más exigente, cual es la historia final que se va a desarrollar, que personajes entran en cada mapa, etc. Esta fase es la más importante pues de aquí saldrá lo que se convertirá finalmente en el juego, y que es comúnmente conocido en las empresas como "*Brainstorming*", ya que se dan ideas generales, de las cuales se seleccionan las principales por su versatilidad de desarrollo y posible inclusión en el juego.

Dentro de la metodología denominada espiral, cada sector que lo compone tiene por propósito facilitar el camino de desarrollo y puesta en escena del mismo proyecto:

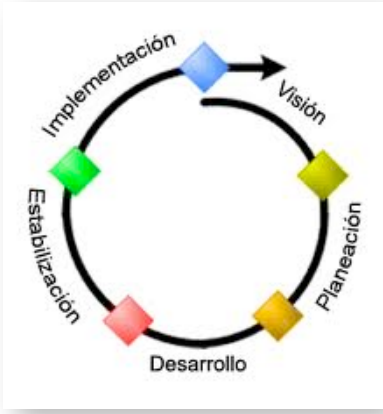


Fig. 3.2. Espiral de trabajo para la creación de proyectos.

El modelo, que a priori puede parecer de lo más sencillo, y lo es, sigue una estructura básica para la creación de cualquier proyecto ya que si se sigue en la medida correcta, puede facilitar la ejecución, pues dispone de las siguientes características:

◉ Las primeras iteraciones, son muy básicas con respecto a lo que se quiere obtener, pero a su vez son también de un coste muy inferior a lo que se espera por el proyecto en su totalidad.

◉ Los pasos que están detallados en la figura, no son necesarios de seguirlos al exactamente, sino que se van adaptando dependiendo de las necesidades que van apareciendo durante el proceso de desarrollo del proyecto.

◉ Se pueden combinar con otros ciclos de trabajo, lo que ofrece una visión más amplia de lo que se está haciendo y como se está haciendo, pudiendo en cualquier momento volver a un paso anterior sin necesidad de dar una vuelta entera.

◉ Lo que realmente le hace útil, es la eficiencia frente a los costes que se producen en cuanto un proyecto se pone en marcha. Con este tipo, se contemplan de mejor manera los riesgos a los que hay que hacer frente.

◉ Y como es una herramienta de uso continuo hasta el final del proyecto, analiza riesgos cada cierto tiempo, y determina en todo momento la viabilidad del desarrollo de alguna parte del trabajo de forma que su coste sea lo menor posible.

3.4. Diseño de la aventura.

Inicialmente se ha de crear lo que verá el jugador en cada momento del juego, es decir, se han de crear las escenas que faciliten el desarrollo de la aventura. Todo ello es necesario para, una vez terminada la implementación gráfica, poder incluirles los diálogos y las preguntas, con la programación basada en JavaScript.

Como se ha indicado anteriormente, hay que estudiar cada mapa y su correspondiente en el entorno de Java que se quiera explicar, por lo que en los siguientes apartados se van a tratar las escenas de cada pantalla, así como una breve introducción a lo que se explica en cada una de ellas.

3.4.1. Inicio del juego.

La primera escena con la que se ha trabajado ha sido el menú de inicio del juego.

Fig. 3.3. Instantánea de la pantalla de inicio.

En ella se han definido las dos partes principales que la componen, las cuales son: menú de inicio y decorado del fondo.

- El menú de inicio, como se puede observar, consta de los botones iniciales y comunes a cualquier tipo de juego. Una vez se pulsa sobre ellos (parte en la que se intercala junto con los Scripts), se accede a la zona especificada, de esta manera, si se pulsa "Nuevo jueguecito" o "Historieta", se carga una pantalla en cada caso distinta, y en el caso de "Continuar", una página en la que se pueden cargar los distintos niveles que componen la aventura.

- La montaña que se ha elegido para poner de fondo, no es más que nada una mera representación de la misma montaña que ha de subir Dante para poder purgar todos sus pecados. Se puede observar los 7 distintos niveles, y como cada uno de ellos se comunica con el siguiente mediante una subida empinada, que emula los escalones en el libro.

Lo único que se puede hacer este escenario, es pinchar sobre el menú deseado y se transporta al usuario a la pantalla que se ha seleccionado. Una vez en el mapa seleccionado se le ofrecen otras posibilidades de interacción con el programa.

3.4.2. Pantalla introductora del juego.

Una vez que se pulsa sobre el botón "Nuevo Jueguecito", se transporta al usuario hacia una pantalla sencilla, pero que indica cual es la situación del problema en el que se encuentra actualmente el protagonista de la aventura.

Fig. 3.4. Carta de Beatriz para Dante.

En esta captura de pantalla se puede observar un poco lo que ya se ha comentado en el capítulo 1 de este trabajo. Dante un alumno que no destaca demasiado entre sus compañeros, está en último año de carrera, con el proyecto y todo presentado, entonces se puede uno preguntar, ¿cuál es el inconveniente de ese alumno?, la solución es sencilla, le queda una asignatura de primero, precisamente Programación de Sistemas.

En este punto entra como en el libro original la inestimable figura de Beatriz, novia de Dante, que le insiste en la importancia de poder terminar para irse con ella y terminar juntos para siempre.

Una vez sabido lo que le espera a nuestro aventurero, es hora de empezar con la historia que nos ocupara durante todo el viaje, el aprendizaje de los principios básicos de la programación, haciendo uso de Java como lenguaje.

4.4.3. Puertas del Purgatorio.

Fig 3.5. Puertas del Purgatorio.

Como se puede observar en la imagen 4.5, el usuario se encuentra abajo a la derecha, y es representado en el juego como una cápsula (personaje por defecto que da Unity). En este mapa se dan los primeros pasos de la aventura de Dante.

En el mapa se pueden encontrar a otros cuatro personajes:

- Virgilio, el cual acompaña a Dante durante todo el viaje.
- Caronte, patrón del barco que llevaba a Dante hasta donde se encontraba Beatriz, desgraciadamente un pinchazo en su barca obliga al héroe a tener que hacer el resto del viaje a pie.
- Catón, que cansado de estar siempre en la puerta que separa el Infierno del Purgatorio, decide visitar a su homólogo el ángel, para pasar un rato más ameno. También representa la figura del que todo lo ha aprobado y menosprecia el esfuerzo de Dante por seguir adelante.
- Y el ángel, el cual al sólo poderle hacer una pregunta de las varias incluidas en su script, el usuario deberá responder acorde a la información que le han dado el resto de personajes.

En esta escena lo que principalmente se intenta, es que el jugador entienda cual es la mecánica del juego. No es necesario, aunque quizá para alguno pueda resultarle útil, tener a mano un bloc de notas, para poder ir apuntando la información más importante y de esa manera no tener que volver a hablar con todos los personajes hasta dar con el que le cuente la respuesta que le ha pedido el ángel.

Cabe destacar una vez más, que las preguntas del ángel son aleatorias, por lo que si en un primer momento había preguntado por el día que era, en la siguiente pregunta, puede ser que pregunte el año en el que se descubrió America, de este modo se aconseja la libreta para tomar datos de relevancia.

Para poder dar una mejor ambientación a los personajes del juego, en este caso y como se podrá ver más adelante, los que son más característicos como en este mapa Caronte (supuestamente un esqueleto forjado con el paso de los años), se le ha adecuado con un personaje "sin rostro", y se le ha realizado un estiramiento de columna, con respecto al resto, y así conseguir una mayor atención del jugador.

3.4.4. La Pereza.

La ambientación que se ha decidido usar para esta escena es una playa, intento de ser paradisíaca, con su puesta de Sol, emulando la relajación que puede producir. Lo que se trata, al ser de pereza general empezar con cualquier cosa que pueda no ser de agrado, se trabaja con el tema de los principios de la orientación a objetos, como se ha de trabajar y cuales son los fundamentos necesarios para sentar una buena base de esos mismos conocimientos.

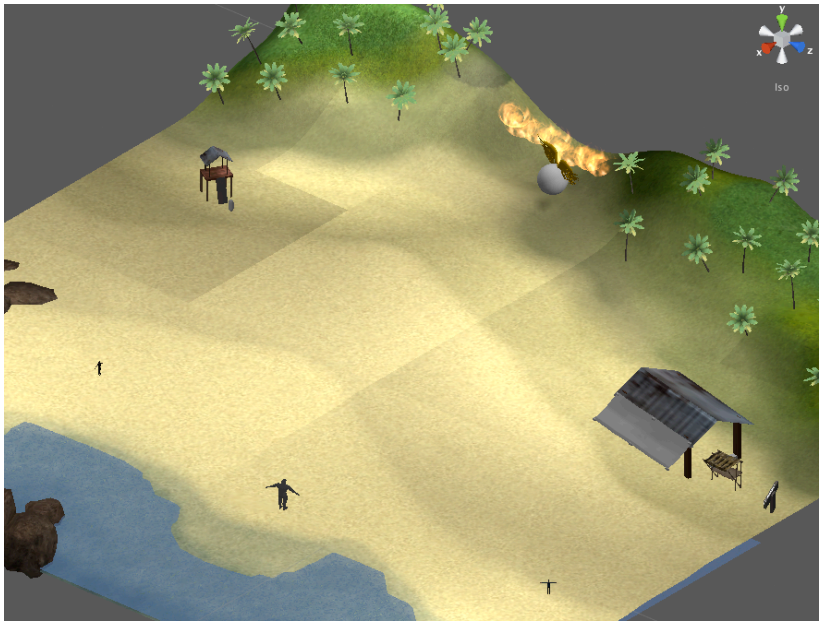


Fig. 3.6. Playa al este del Purgatorio.

Tal como se ve en la imagen, se representa la playa, con su "chiringuito" típico español, el puesto de vigilancia, de estilo "vigilantes de la playa", incluido por normas de seguridad. En esta escena aparecen los siguientes individuos:

- Homer Simpson, perezoso reconocido mundialmente, pasa las horas muertas en este lugar, lejos de su familia y sobre todo de su trabajo. Indica al usuario cuales son las características principales de la orientación a objetos, y da una breve explicación de cada uno de ellos.

- Perezoso 1, chaval tipo de las playas que no hace nada, y se pasa el día tomando el Sol y bebiendo refrescos. Incide en la importancia de saber como se pueden crear objetos de la

misma clase en la que se encuentra el programa, para un uso eficiente del código, así como la creación de referencias para poder trabajar con ellas.

- Perezoso 2, explica las diferencias que existen entre los atributos de clase y los de instancia, y los tipo de métodos que se pueden hacer uso dependiendo de la función que se quiere realizar con ellos: si se quiere devolver algún dato, los que imprimen los datos por pantalla o simplemente los que cambian el estado de los atributos creados en los objetos.

- Perezoso 3, otro que está todo el día sin hacer nada, y que lo único que hace es molestar a los del chiringuito con sus tonterías e idas de cabeza. Da una breve explicación en cuanto a las excepciones se refiere, dando importancia a que es bastante recomendable comprobar los tipos de los atributos con los que se va a trabajar antes de ponerse a operar con ellos, pues pueden contener problemas, que durante la ejecución pueden ser difíciles de reconocer.

- Y el ángel, que como se acabará dando cuenta el usuario, deberá poner a prueba si ciertamente a comprendido o asimilado de forma positiva las conversaciones que ha tenido con el resto de los personajes.

3.4.5. La Gula.

Inicialmente para este tipo de escena, se decidió incluir una escena de la hora del té del libro Alicia en el País de las maravillas [33], en el cual se podía imaginar una mesa repleta de platos, tazas y teteras, junto con un bosque cercano rodeado de árboles y setas gigantes. La idea final que se planteó era realizar una larga mesa situada en el centro, con una gran seta en el centro, emulando dicha escena.

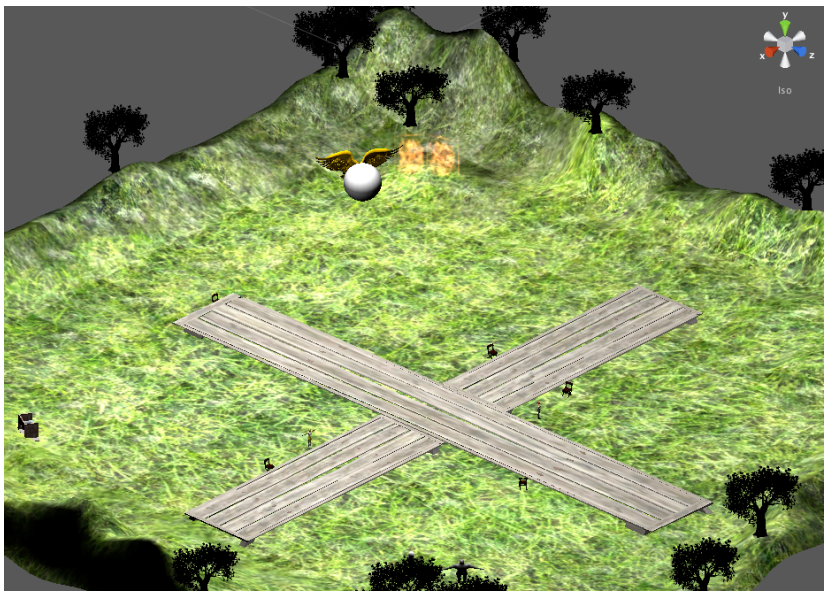


Fig. 3.7. Emulación de la imagen de la hora del te

En esta escena, se continua con una breve introducción de la orientación a objetos pero haciendo más hincapié en el tema, dando unas pautas más profundas sobre: la ligadura dinámica, en cuanto a la palabra "*static*" y sus funciones cuando se quiere hacer referencia tanto a un atributo como a un método; como funcionan las clases abstractas, viendo cual es el tipo de herencia que se produce cuando h es un hijo de p y como el hijo ha de declarar los métodos que posee el padre de una forma

especial; o las interfaces, las cuales permiten una de las características principales de la programación orientada a objetos, el polimorfismo.

Los personajes principales en esta escena son:

- Obelix, conocido personaje [34] indica las primeras nociones sobre la ligadura dinámica y de como se han de usar las llamadas entre los métodos de las clases de forma más sencilla.

- Goloso 1, personaje que representa el perfecto glotón de comer hasta reventar o ahogarse (como es su caso), explica de forma superficial como se han de realizar las referencias entre las clases abstractas, con el h hijo de p, y como los métodos del padre pueden ser heredados por el hijo, o en caso contrario, el hijo no desea heredar más que un método de su patriarca.

- Golosa 2, en este personaje se tiene al caso contrario del Goloso 1, el cual es mucho más cercano a lo que Dante relata en la "Divina comedia". Golosa no come nada y le repugna la gente que come como si de la última cena se tratase. Sin embargo, lo que trata es bastante usado en la programación avanzada, las interfaces, en donde cuando un hijo quiere heredar algún método de otra clase que no es superior, necesita hacer referencia a la interfaz que desea heredar.

- El ángel, en la escena de la Gula, se obliga especialmente al usuario a poner especial atención a lo que pregunta, ya que en alguna de las preguntas que realiza, es necesario poner la respuesta en inglés para poder superar el nivel.

Como se va a comprobar a partir de este punto, la parte básica de la orientación de objetos en Java se deja de usar y se ha pasado a lo que realmente interesa, la programación de estructuras de datos y más compleja.

3.4.6. Orgulloso de ser.

Supuso un grave dilema que poner asociado a este pecado, puesto ¿qué se supone que es orgullo? La ambientación que se ha adoptado para la escena, es la del monte del Olimpo, ya que los Dioses de aquella época, eran orgullosos, con lo que podían hacer sobre los mortales y por eso se les consideraba seres sin corazón en la mayoría de los casos.

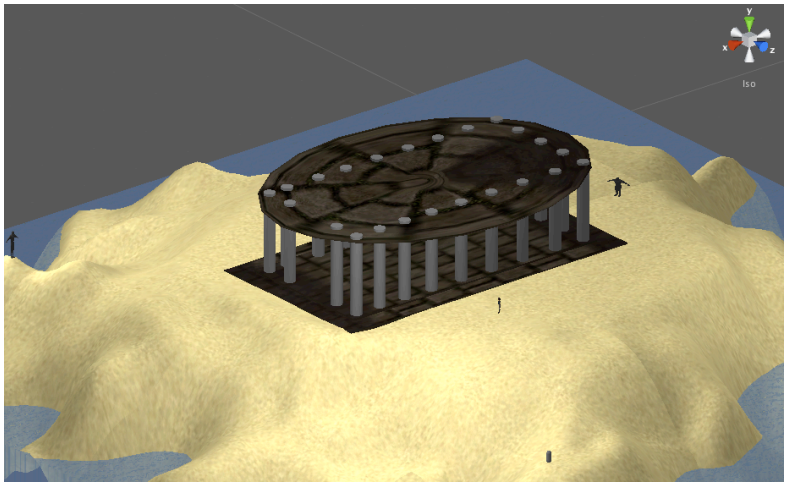


Fig. 3.8. Representación del Partenón.

El tema tratado en este escenario ha sido la introducción a los alumnos a las Listas Enlazadas, y a como funcionan para poder trabajar de una manera eficiente con ellas. Se les familiariza con las listas, que de forma similar son como arrays, pero con una ventaja, se pueden hacer tan grande o pequeñas como se desee, puesto que no tienen un tamaño prefijado como aquellos. Pero también tienen un inconveniente, para poder buscar un dato dentro de las mismas, es necesario recorrerlas desde la primera posición hasta el final, haciendo la búsqueda más lenta.

Los personajes que explican su vida, y algo más son:

- House, personaje carismático donde los haya, nos introduce con su sarcasmo y su forma de hablar en lo que a listas se refiere. Hace hincapié en como es su funcionamiento, y cual es el problema de hacer un uso excesivo de ellas. Aunque por otra parte también indica cuales son los beneficios de su uso.

- Orgulloso 1, persona que se ve a lo largo de la vida, que lo único que le importa es ser el mejor en todo y ser el más rápido, y sin importarle que ha de hacer para llegar a conseguirlo. Este caso se ve mucho durante los años de la carrera, en la cual la importancia que se da a la nota es lo principal para algunas personas. Indica el uso de la referencia “NULL”, y como se ha de hacer uso en las listas enlazadas cuando sólo se tiene un objeto, o varios.

- Orgulloso 2, toma el pelo a Dante ya que no ha sido capaz de aprobar la asignatura aún. Mostrará la ampliación que se le dan a las listas enlazadas, con las doblemente enlazadas. De misma estructura que las simples, lo único que las diferencia es que los nodos de las simples saben quien va detrás de ellos, pero que en el caso de las dobles, se sabe quien va delante y quien va detrás de cada una de ellas.

- Homer (otra vez), poner a este personaje en la escena no fue decisión de que es orgulloso, pero sí que se trata de una frase que dice en una capítulo de la serie que es muy preciso para poder explicar como se extrae y como se inserten los nodos en las colas dobles.

- El ángel, representación divina de los dioses de la Grecia antigua, realiza las preguntas pertinentes sobre la escena desde el punto que se puede dar a alguien soberbio.

3.4.7. Un pequeño pedazo de Amsterdam.

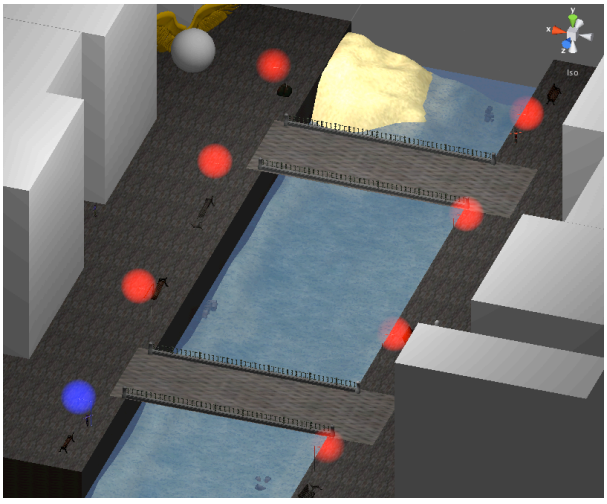


Fig. 3.9. Un pequeño trozo de Amsterdam.

La escena de la figura 3.9, representa un pedazo de la conocida ciudad de Amsterdam. Y ¿por qué este escenario? Bien, la verdad es que el temario que se explica en este lugar, tiene que ver con la funcionalidad de las colas, las pilas, y como de ellas se pueden extraer e insertar nuevos datos.

Haciendo un breve resumen, las colas en programación, son las que trabajan de la forma FIFO, First In First Out, es decir, las primera que entra en el tubo (el tubo es la mejor manera de ver el desarrollo de ambas aplicaciones), son las primeras que salen del tubo, como en la colas de espera de las tiendas de telefonía, el primero que llega es el primero que es atendido. Mientras que en las estructuras LIFO, Last In First Out, el último dato que entra es la primera que sale, se podría imaginar como una pila de platos, todos ellos de la misma forma, los primeros que se ponen en lo alto de la pila, son los primeros escogidos en caso de ser necesarios. Un ejemplo sencillo es el mostrado en la siguiente figura:

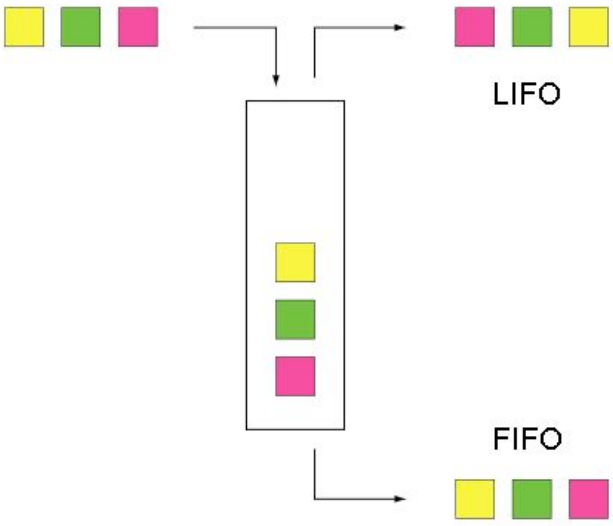


Fig. 3.10. Explicación implícita de las características FIFO y LIFO.

Lo más destacado en esta escena, a parte de que se trata de la única escena en la que la programación es distinta al resto, es obligatorio hablar con todos los personajes de la escena para poder optar a la pregunta del ángel:

- Lujuriosa 1, como ya se ha indicado anteriormente, cada uno de los personajes va a tratar de forma específica un tema relacionado con las colas y con las pilas. Esta señorita nos indica cual es el funcionamiento principal de las pilas, y cual es la principal diferencia con respecto a las colas.

- Lujurioso 2: siguiendo con el tema de las pilas, este ladronzuelo de poca monta nos ofrece saber información a cambio de nada. Su papel está estrechamente relacionado a su trabajo, la relación de poder meter y sacar objetos donde le interesa, así pues, da especial importancia a *push* y *pop*.

- Lujurios@ 3, la leyenda dice que las luces rojas pertenecen a las mujeres y que las azules... La presente señorita, nos enseña la forma con la que se han de trabajar las colas en programación. Realizando el símil de las colas del supermercado para facilitar la comprensión de su funcionamiento a los jugadores.

- Lujurioso 4, de la misma manera que el Lujurioso 2 nos introduce en los conceptos básicos de como se ha trabajar extrayendo e incluyendo los datos para las pilas, este no podía ser menos, y nos indica de manera totalmente gratuita las mismas pautas, pero en este caso para las colas, con *dequeue* y *enqueue*.

- El ángel, bastante más lanzado a lo que se está acostumbrado desde que se le conoce en las Puertas del Purgatorio.

Como se ha indicado en un par de párrafos anteriores, es la única escena distinta. En este caso se ha decido crear un diagrama de conversación más intrincado, para poder hacer que los usuarios hablen con todos los personajes y escuchen que es lo que cada uno de ellos ha de decir. Una vez que se ha terminado de hablar con los distintos personajes, cada uno de ellos dará un objeto distinto, pero en un orden establecido con anterioridad para poder seguir un ritmo de conversación adecuado a lo que cada personaje cuenta.

3.4.8. La envidia corroe por dentro.

Al igual que lo sucedido en el mapa del Orgullo, a esta escena era difícil de poder atribuirle un lugar como tal para que se diese a entender de una manera más sencilla a que pecado se está refiriendo. Un teatro es la opción más destacada, también se tuvieron en cuenta una sala de reuniones, un pase de modelos o un cementerio, pero que no llegaron a hacer la mella esperada como con el teatro.

En este mapa, lo principal es ser el protagonista de la obra, pero sólo puede haber uno, y el resto ha de mirar como el más inepto ocupa el puesto más preciado por los que se esfuerzan. De esta manera se ha decidido que el tema que se ha de tratar sean los árboles de programación, con una breve explicación más a fondo en cuanto a los Nodos, y finalmente una breve descripción de como se pueden recorrer los árboles.

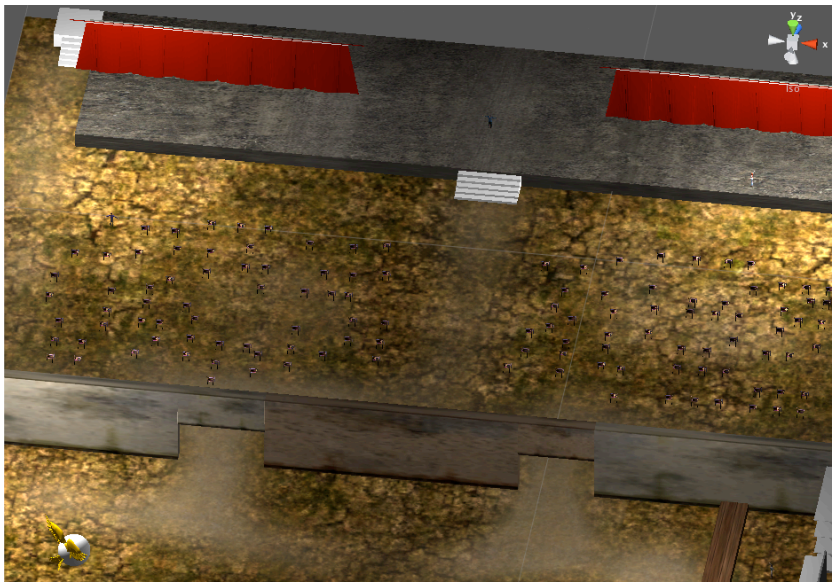


Fig. 3.11. Teatro Kasper-Lolkat.

La presencia de los personajes, es tan molesta al resto que se ha intentado plasmar ese malestar entre los presentes de forma, eso si, siempre formal:

- Envidioso 1, marginado del resto de los compañeros, es el candidato perfecto para poder introducir como funcionan los árboles de la programación, ya que como él indica en la breve conversación, desea poder tener un p del que poder heredar, y unos h a los que dejar un legado de métodos y atributos. Todo lo anterior, junto con una breve indicación sobre lo que la asignación NULL significa para los árboles.

- Envidioso 2, protagonista de la obra que se representa, demuestra que no siempre al que se tiene envidia es el mejor en lo que se pensaba. Lo que el susodicho explica, es como se han de colocar las hojas dentro del árbol, para poder hacer después un buen recorrido de búsqueda, pero ¡¡¡¡OJO!!!!, la ordenación que se hace es la de la clave de cada nodo y no del valor que contiene cada nodo dentro de sí, ya que puede contener cualquier tipo de dato.

- Envidiosa 3, para no perder la práctica, criticar es lo que mejor hacen los envidiosos y en este caso no hace nada más que empeorar la relación. Introduce un poco más el funcionamiento de los nodos de los árboles y de las operaciones que se pueden hacer en ellos, para poder sacar información de lo que en ellos se encuentra, a esto se le conocerá durante el estudio de la asignatura como Árbol de jerarquía.

- Envidiosa 4, marginada de los ensayos y relegada a un puesto de vendedora de entradas para la obra, pide desesperadamente información de como va desarrollándose la obra, puesto que nadie le comunica nada. El lado positivo de estar tan aburrida, le beneficia, puesto que es la encargada de explicar como funcionan los recorridos de los árboles para una mayor comprensión a la hora de decidir que tipo de búsqueda se quiere realizar.

- El ángel, en esta escena hace preguntas más complejas de lo que está acostumbrado el usuario, por lo que es un requisito imprescindible, el haber hablado con todos los personajes de la escena.

3.4.9. Los iracundos.

Una vez se ha visto el tema de los árboles en el escenario anterior, es hora de ahondar un poco más en el tema de la búsqueda dentro de los mismos. La trama se desarrolla alrededor de un ring de boxeo. La forma en que se puede trabajar con los árboles de una forma más rápida es con la extracción, inserción y búsqueda de datos, como ya se ha citado anteriormente.

Es bastante importante explicar de forma muy concisa como trabajan los árboles y cuales pueden llegar a ser su utilidad en el ámbito de la programación. De esta manera el pecado elegido para este tipo de tema, ha sido la Ira, ya que la primera vez que se tratan estos temas, no es sencillo asimilarlo y es necesario una segunda revisión para poder comprender de una mejor manera lo explicado.

Fig. 3.12. Pelea verbal para el escenario de la Ira.

En este cuadrilátero la pelea no es física, aunque lo parezca sólo en forma, pues lo que se ha de hacer dentro de él, es pelear verbalmente. Se da un tema para tratar y se ha de defender o explicarlo de la forma que uno conoce, donde el tema son los Árboles Binarios de Búsqueda:

- Irascible 1, comentarista del combate, obliga a meterse al protagonista en una pelea que no había buscado en ningún momento. Como se ha visto hasta ahora, el primero de los personajes da las nociones sobre el tema que se trata dentro de cada escena, y en este caso se habla de los árboles binarios de búsqueda. Todo lo que se explica, se ha de haber visto en los escenarios anteriores ya que es una mera explicación básica de lo que se hace en ellos.

- Irascible 2, entrenador personal de la sin hueso, la importancia que representa dicho personaje, está en la búsqueda de los nodos deseados dentro del árbol jerárquico en cuestión. Todo lo que se ha visto, se está viendo y se va a ver, tiene por finalidad aclarar que lo que se compara siempre es la clave que se le asocia a cada nodo, y no la referencia que tiene dentro del mismo.

- Irascible 3, como en todo evento que se precie, siempre existen personas que desean hacerse ricas a partir de otras, y en este caso no iba a ser una excepción. El personaje hace una breve referencia a las tres posibilidades que existen a la hora de insertar un nodo, y la forma que se ha de hacer para no tener ningún tipo de sorpresa a la hora de recolectar la información una vez realizada la operación.

- Irascible 4, y como no existe disputa en la que no haya un juez, este comentará cual es su tema favorito dentro de los árboles y para ello relata como se han de eliminar (los nodos, no los adversarios) de cada árbol. Con los ejemplos que propone, se espera arrojar cierta luz sobre como se ha de hacer y que puede pasar si no se hace un guardado de los datos antes de que se realicen cualquiera de los datos.

- El ángel irascible, la batería de preguntas aumenta y con ello las posibilidades de respuesta que se pueden dar. Se incluyen aclaraciones en las mismas preguntas para evitar la posible confusión de los alumnos, pero se les hace pensar sobre la pregunta antes de responder.

3.4.10. La Avaricia no es buena para nadie.

En el último mapa de esta herramienta, se ha intentado plasmar un poco más lo que se puede ver en un juego de tipo comercial, aunque por otra parte, se ha tenido que eliminar algún elemento para no hacer demasiado pesada la aplicación.

El último tema que se ve es bastante simple, pero que tiene una gran importancia a la hora de programar. Se trata de los algoritmos de búsqueda dentro de los programas de ejecución. Dichos algoritmos deciden si un programa es o no lento durante su ejecución.

Existen una infinidad de algoritmos de búsqueda, pero en este caso, se ha decidido no tratar todos y centrarse sólo en los más representativos con vistas a la utilidad que se pueden dar dentro del ámbito universitario. De esta manera se trata la diferencia que existe entre las búsquedas de tipo lineal y binaria, y las dos herramientas que más se usan en cuanto a la ordenación de datos en los sistemas: ordenamiento por inserción, y los de selección.

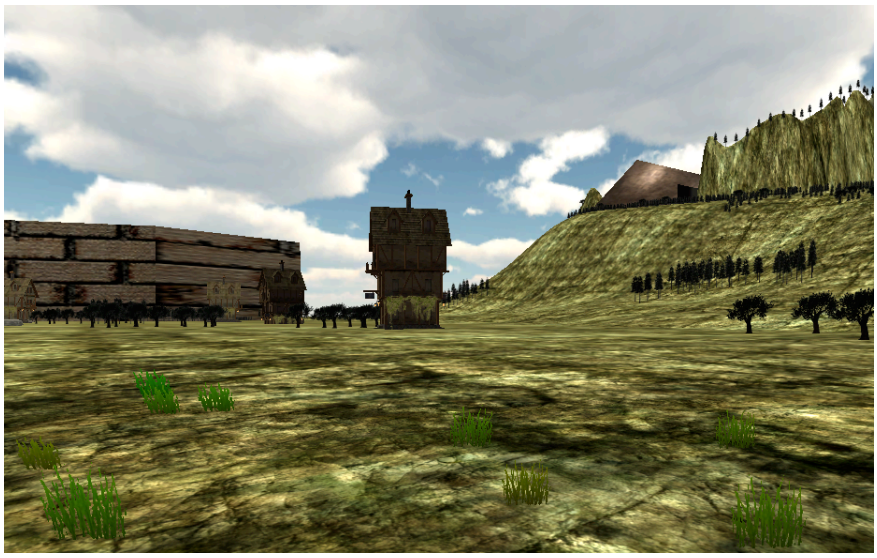


Fig. 3.13. Visión en primera persona de la escena Avaricia.

Los personajes que habitan este mapa, se les conoce como avaros, ya que cada uno de ellos tiene un bien preciado o muchos, y no son capaces de desprenderse de ello ni para prestarlo. Se ha colocado una montaña en una parte del mapa, para ejemplificar la soberanía del que tiene el poder de tener todo, y dentro de la misma, un personaje bien conocido por todos:

- Tío Gilito, según un estudio realizado todos los años por la revista Forbes, es el personaje ficticio más rico de todos los tiempos. Aunque está en lo alto de la colina, rodeado de dinero, ayuda al usuario a poder entender donde está, y a recordarle para que ha ido tan lejos en su viaje, y ahora que ha llegado al final, no se puede detener en su esfuerzo por aprobar. Prepara a Dante ante el nuevo tema que se presenta, los algoritmos de búsqueda, ya que hasta el momento no se había visto nada de este estilo en ninguna escena anterior.

- Gollum, como personaje, se supone que es bastante conocido a día de hoy por casi todo el mundo, pero ¿y cómo avaro? En este caso se expone de nuevo la importancia que tiene su anillo y como no es posible quitárselo de ninguna manera. Ya que es bien sabido que tiene doble personalidad, Gollum y Sméagol, serán los encargados de explicar las diferencias entre el tipo de búsqueda lineal y la búsqueda binaria.

- Avaro 3, la personalidad de la persona avariciosa no le permite poder dejar las cosas, y mucho menos desprenderse de ellas, por eso este personaje tiene miedo de que vengan a quitarle todas sus posesiones. Su papel en la historia es contar como trabaja el ordenamiento por selección, con un ejemplo sencillo, y una pequeña ayuda a la hora de aprender a usarlo en el campo de la programación.

- Avaro 4, la finalidad de este personaje es molestar a avaro 3, ya que tiene todo ordenado de una manera que no es de su agrado, y por lo tanto le molesta que lo haga. El tema que trata es también el del ordenamiento por inserción con un ejemplo sencillo, y una manera fácil de entender en caso de duda.

- El ángel, por tratarse del último mapa en el que se va a desarrollar la historia, es importante poner a prueba los conocimientos que ha adquirido el alumno, por lo que en este caso, en ningún momento ninguno de los personajes cuenta la solución a la pregunta del ángel, por lo tanto, el usuario debe de saber en que orden se le pide que coloque los valores, y sólo si están bien ordenados, podrá terminar el juego.

La decisión de que el mapa tuviese el aspecto de un pueblecito, viene de la idea que se tiene de la edad media, en la cual los señores vivían rodeados de riquezas, mientras el pueblo llano tenía que lidiar todos los días con asaltantes, y personajes de baja calaña. Este mapa explica perfectamente esto, en un entorno más sencillo. Tío Gilito, es el señor feudal, y el resto sus jornaleros.

3.4.11. ¿Fin de la historia?

El final de la historia no es otro que el que se muestra en la siguiente figura:

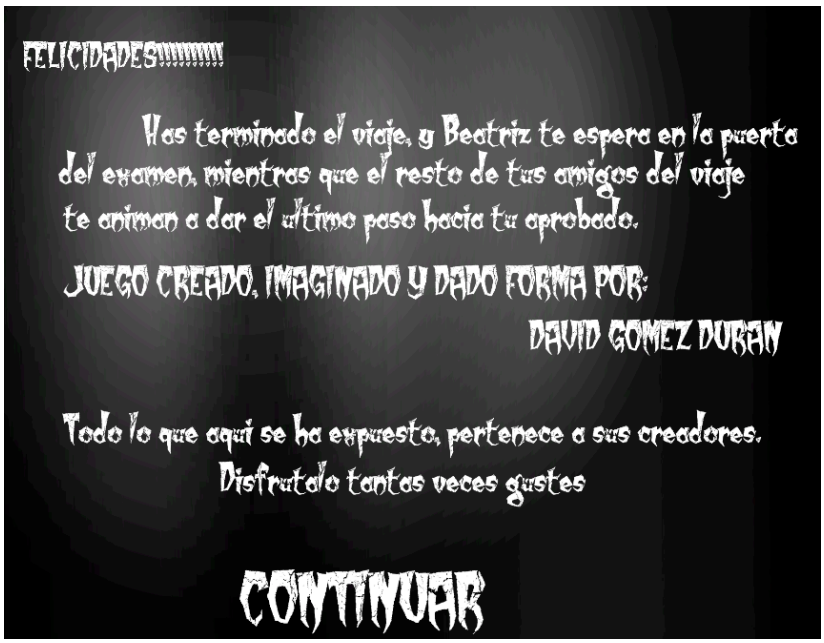


Fig. 3.14. Final del juego o ¿inicio de otro?

Como se observa, el final del viaje nos espera en la puerta del aula del examen con Beatriz delante de ella. La idea se da, puesto que una vez pasado todo el purgatorio, Dante en las puertas del paraíso ha de beber de los ríos que le borran todo pecado y le devuelven todos los recuerdo (como se ha comentado en el capítulo 3.2), y empieza el viaje en el paraíso guiado de su amada. Eso

mismo se ha pretendido en este final, una vez terminado el juego, se suponen asimilados los conceptos más básicos de la asignatura, que junto con la ayuda del profesor, y algún que otro ejercicio más complejo, ayudará a superar la asignatura de una forma más sencilla.

Con todo lo que se ha tratado durante todo el juego, no se espera que se trate del final del mismo, más bien se espera que se trate de una herramienta de ampliación, aunque ya se ha comentado anteriormente, el alumno ha sentado sus bases, y deberá seguir gracias al apoyo de su guía en la vida real, el profesor.

Capítulo 4

Diseño e implementación

En el capítulo siguiente se puede encontrar una explicación más detallada sobre como se ha desarrollado, diseñado e implementado la herramienta en su totalidad.

El primer punto que se trata es el requisito con el que se ha hecho frente a lo que se ha necesitado crear para poder satisfacer al usuario. Así pues, la creación de esta herramienta es fundamentalmente una fuente de apoyo a los alumnos en los primeros pasos en cuanto a la programación de objetos se refiere.

Posteriormente, se mostrará el diseño del proyecto, como se ha organizado estructuralmente y los scripts que se han creado, para poder tener un desarrollo correcto de la aventura, para finalmente, desarrollar las estructuras de programación usadas.

Cabe destacar en este apartado, que la creación de las escenas ha sido en todos los casos igual, excepto en la Lujuria, la cual ha sido y es la escena principal de este trabajo. La razón de haber trabajado este mapa más exhaustivamente es la importancia del tema que se explica en el mismo. Por tanto, la demostración de esta escena es la más representativa dentro de toda la aventura.

Una vez explicadas las capas más superficiales, las cuales corresponden a una gran parte de la visualización del juego, se hace frente a la implementación del código usado en JavaScript. Este apartado, pretende abordar las partes no visibles del trabajo realizado. Este tipo de trabajo es invisible para los usuarios, y por tanto no pueden acceder a la programación usada en el mismo.

4.1. Requisitos del usuario.

La principal motivación para la creación de este proyecto, fue la de poder favorecer el estudio de la asignatura desde un entorno dinámico.

Este proyecto crea un mundo virtual, en el que la sucesión de los temas tratados hace más sencillo el poder entender los distintos módulos que Java ofrece. El perfil que se busca entre los usuarios, es el del no haber trabajado anteriormente con Java en lo que a herencia, excepciones o listas enlazadas se trata. Ahora bien, es necesario el poder saber las bases de la programación básica, ya que sin esa base, el usuario no podrá asimilar de forma didáctica lo que se explica.

Cada una de las escenas ofrece al usuario la información básica sobre un tema de la asignatura determinado, como se pudo ver en el Capítulo 3. De esta forma queda estructurado el desarrollo de la asignatura con el desarrollo de la historia, empezando desde la base de la Programación Orientada a Objetos.

Durante el desarrollo del juego, se tuvo presente en todo momento, la creación de una herramienta unipersonal, sin necesidad de ser gestionada por una conexión en red. Cada usuario puede acceder a los contenidos que desee, siempre que lo necesite desde el menú principal, por lo que hace de esta herramienta, una guía interactiva.

Según se ha comentado en el párrafo anterior, esto pretende ser una guía interactiva, pero no “LA” guía definitiva de la programación en Java. La explicación, así como los ejemplos y una mayor profundización de la asignatura se delega al cuerpo docente, el cual, puede hacer uso de este trabajo.

4.2. Diseño del sistema.

Como se ha explicado en capítulos anteriores, Unity3D es un mero editor de mundos virtuales, que permite diseñar y dar forma a lo que la mente humana de cabida. Pero por sí mismo no es capaz de crear programas ejecutables, entendiendo programas ejecutables aquellos que pueden crear conversaciones u acciones determinadas para personajes de la escena, inclusive el propio jugador.

Con la herramienta que acompaña a este editor, MonoDevelop-Unity, es posible dar vida a ese mundo creado ya sea desde cero o añadiendo nuevas funciones a algo ya creado.

En un primer momento, se hizo uso de una matriz igual para todas las estructuras de diálogos, lo que provocaba que el juego se convertiese en una simple: acción y reacción por parte del jugador, tanto a la hora de hablar con los PNJ's, como a la hora de tener que responder la pregunta del ángel.

Por esa misma razón se pensó que al menos una de las escenas tendría que aportar algo distinto a este proyecto, y esa fue la razón por la que se diseñó específicamente la escena de la Lujuria de manera distinta al resto.

Para facilitar la comprensión de las diferencias entre Lujuria y el resto, en cada punto que se exponga, se hablará primero del caso general (es decir el de todas las escenas) y posteriormente se hará un mayor hincapié a dicha escena en particular.

En este apartado se van a desarrollar tres puntos troncales asociados a la creación de cualquier juego:

- **Dinámica general de las escenas**. Como funcionan los niveles, y cuales son los scripts usados para poder hacer interactivos a los mismos.

- **Interfaz de usuario**. Como ha sido el desarrollo de las conversaciones, como son las interacciones con los PNJ y como se ha creado la interfaz específica para la escena de la Lujuria.

- **Estructura de programación**. En este apartado se muestra con mayor profundidad como es el flujo de información entre los distintos scripts, y cuales son las funciones principales de las mismas.

4.2.1. Dinámica general de las escenas.

Hasta el momento sólo se ha mostrado lo que visualmente es perceptible para el usuario en general, pero en este apartado se van a comentar los entresijos del juego. En el Capítulo 3, se introdujo una breve guía informativa de como estaba dividido cada una de las escenas, y cuales eran los temas a tratar en cada uno, si bien no se introdujo específicamente en como funcionaban dichas escenas, si se indicó que trataban.

En este proyecto, se ha hecho uso del lenguaje JavaScript, por la facilidad que resulta trabajar con él a la hora de programar, pues el uso de sentencias y codificación es similar a Java.

La estructuración de las escenas es sencilla, en cada una existe un número determinado de PNJ's, con los cuales se ha de hablar para poder saber que te cuentan y tener una breve base sobre lo que el ángel puede preguntar.

En todas las escenas se puede ir directamente a hablar con el ángel y superar la prueba, o como se ha comentado hablar primero con todos. Pero no es así en todas las escenas, ya que la escena de la Lujuria es especial, en ella se ha programado la mayor parte de los scripts ejecutables con dependencia de lo que haga el usuario, obligándole a tener que hablar con cada personaje, en un orden determinado para poder dirigirse al ángel y contestarle.

Este punto va a tratar más detalladamente de cuales son los scripts creados, y cual es su finalidad, para posteriormente profundizar más en como se comunican y funcionan, para una mayor comprensión de lo que ocurre durante el juego.

La primera pantalla que aparece una vez ejecutada la aplicación es la del Menú de inicio. Este menú ofrece las distintas posibilidades a los usuarios de forma que se elija cual de todas desea ejecutar. Como ya se pudo ver en la imagen 3.3, del capítulo anterior, dispone de varias opciones, donde cada una de las cuales es un script independiente con respecto al resto.

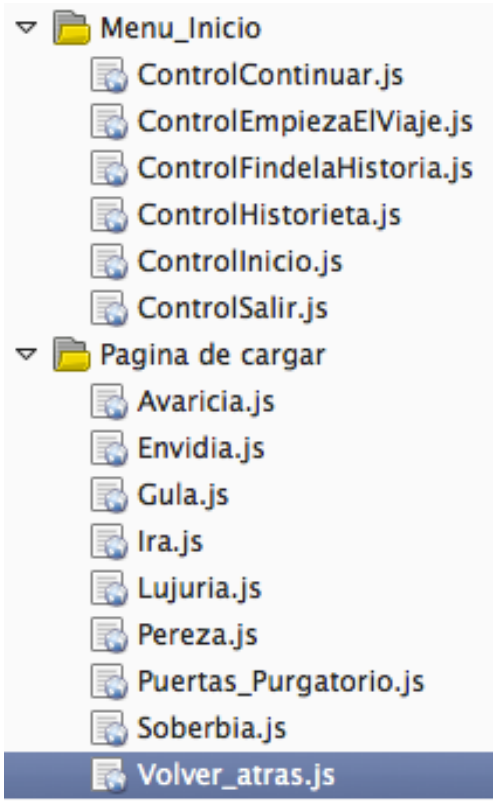


Fig. 4.1. Menú de inicio del juego y página de cargar.

Cada uno de los sub-menús creados dentro del menú inicial, es un script específico. Esto favorece la facilidad a la hora poder hacer cualquier cambio dentro de cada uno de ellos, sin tener que alterar el funcionamiento de los otros:

- Control salir. Este script es de los más sencillos, ya que su función es la de cerrar la aplicación y volver al escritorio del equipo.

- Control inicio. En este caso, el script está configurado para que una vez que se inicie el juego, se transporte al usuario a una breve historia sobre lo que va a tratar el juego, y cual es la finalidad del mismo.

- Control empieza el viaje. Una vez que ya se expone al jugador en que consiste la historia, se le dirige al primero de los mapa, en donde comenzará a explorar la escena y las conversaciones con los PNJ.

- Control de la historieta. Se transporta al usuario a una escena, que al igual que en el caso anterior es una breve introducción al porque de Dante en este viaje.

- Control fin de la historieta. Una vez se ha pasado el juego, se conduce al usuario de nuevo al menú de inicio para, en caso de desearlo, volver a pasarse la aventura.

- Control continuar. Para la creación de un sistema de carga sencillo se creó este script de acceso rápido a las escenas creadas y su posible carga.

Haciendo referencia al último de los scripts citados anteriormente, ControlContinuar, es relevante a la hora de poder acceder a la escena deseada sin tener necesidad de repetir toda la historia.

Una vez que se entra en el juego, cada escena es un mapa distinto en el cual los diálogos son lineales y no es necesario tener que guardar ningún objeto para poder seguir la aventura o poder continuar con las conversaciones como en el caso de la Lujuria.

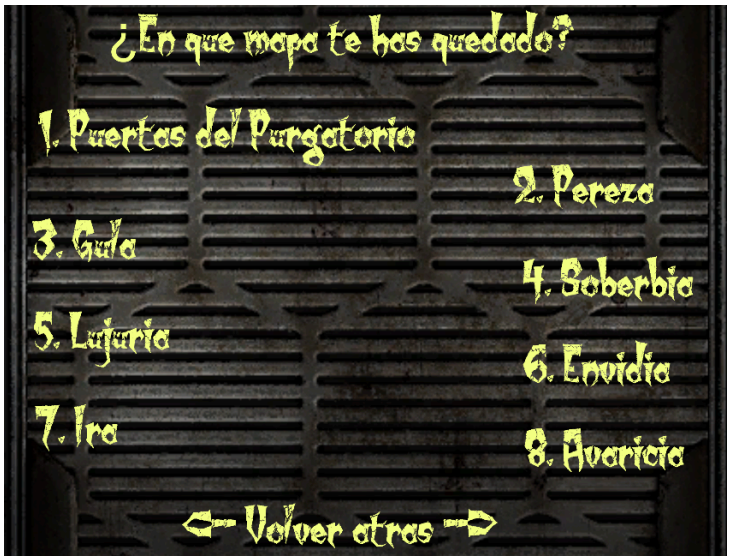


Fig. 4.2. Escena de carga de pantallas.

Durante el proceso de creación del juego, se diseñó una forma de poder acceder a las escenas de la manera más rápida y sencilla posible. Una vez que el usuario accede a esta opción, se le ofrece poder empezar en cualquier escena del juego, pero ¿puede acceder a la última escena sin haberse pasado todas las anteriores escenas? La respuesta es “No”.

Se ha decidido, que para poder entrar en la escena deseada, se ha de hacer mediante una serie de códigos facilitados durante la aventura. De esta manera, una vez terminada una escena, se obtiene el código de la siguiente sin necesidad de tener que empezar desde la escena primera, o acceder sin habérselo pasado antes.

¿Y por qué no se programó una opción de guardar y cargar? Este tema se ha tratado minuciosamente para añadirlo o no a la aventura. Por tanto, se van a exponer las ventajas e inconvenientes de no haberlo creado:

- **VENTAJAS**.
    - No es necesario tener que guardar nada de información dentro del ordenador de cada usuario.
    - El juego puede funcionar en cualquier plataforma, no así la forma de poder cargar y guardar los datos. En Windows es más sencillo de guardar los datos deseados que frente a Macintosh, el cual es mucho más reservado.
    - Como se ha indicado, las escenas son lineales, por lo tanto no es necesario guardar ningún estado especial del jugador.
    - El usuario, puede acceder en cualquier momento a cualquier escena. Esto es decisivo, a la hora de que cada uno puede elegir cual es el tema que desea repetir.

- **INCONVENIENTES**.
    - La única escena que tendría que guardar el estado del jugador es la Lujuria, por lo que si el jugador deseara salir de la misma en cualquier momento, a la hora de volver acceder, tendría que volver a hablar con todos los PNJ's para poder concluir en la pregunta del ángel.
    - Si el jugador pierde las contraseñas, tiene que volver a empezar desde el principio para poder disponer de ellas de nuevo.

Evidentemente, esto plantea una situación problemática: si el jugador está en mitad de una escena y ha de abandonarla repentinamente, no se guarda su avance, ni los personajes con los que ha hablado. Pero no es del todo posible que un jugador abandone en mitad de la partida, sabiendo que si la supera, se le otorgarán los códigos para no tener que volver a repetirla.

Ahora se desarrollará el diseño de los programas usados en las escenas, generalizando primeramente en los aspectos generales, y concluyendo con la escena específica referente a las colas y pilas.

Para poder tener una buena estructuración del proyecto, y un fácil acceso a los archivos de cada escena, se crearon distintas carpetas, dependiendo de cada mapa, y en cada una de ellas se guardaron los archivos correspondientes, todos ellos escritos en JavaScript, y cuyo formato es *.*js*.

Cada escena dispone de unos archivos creados para su propio uso, de esta manera el personaje sólo interactúa con los personajes de esa escena, sin hacer saltar los scripts de los otros mapas.

Fig. 4.3. Carpeta específica para la escena Gula.

Generalmente, todas las escenas tienen la misma estructura: Un menú de la escena y los distintos textos creados para los personajes de la escena. De esta manera, la interacción del personaje principal con el resto de PNJ's es del todo lineal.

Veamos más detalladamente como está creado cada uno de los scripts:

- Scripts de conversación lineal. Estos archivos están creados para contener únicamente información relacionada con las conversaciones que se van a desarrollar entre el jugador y los personajes de cada escena. En el caso expuesto de la figura 4.4, serían los relacionados con los personajes: *Goloso_1*, *Goloso_2*, y *Obelix*.

- Script del ángel. Este programa es específico únicamente para cada ángel de la escena. Dentro, se encuentran las preguntas y las respuestas que cada uno de ellos realiza a los usuarios, y es él quien decide si deja pasar de nivel o, por el contrario, ha de repetir la escena para poder comprender mejor los conceptos explicados. Dentro de cada uno de estos, existe un número indeterminado de preguntas, así como de respuestas, lo que permite que cada vez que se hable con él, las preguntas, y las propias respuestas varían con respecto a la expuesta anteriormente.

- Y finalmente, el Menú de pausa. Cada menú es individual en cada uno de los niveles creados. Dentro, existen varias opciones que plantean a los usuarios que desean hacer durante la partida. En apartados posteriores se explicará más detenidamente como funcionan y cuales son sus principales características.

Cada uno de ellos se activa cada vez que el usuario se acerca a cualquiera de ellos, de tal manera que no activa los otros ejecutables de la misma escena o de cualquier otra.

En todos los pecados el diseño es el mismo, el usuario tiene que habla con el resto de PNJ´s, y sólo podrá pasar de nivel en el caso de responder correctamente al ángel. Pero en el caso de la Lujuria, este principio no se cumple.

El mapa de la Lujuria, tiene un desarrollo, en cuanto a conversación lineal, al igual que los otros, pero en este caso, se ofrece al usuario un abanico de posibilidades con respuestas de tipo múltiple.

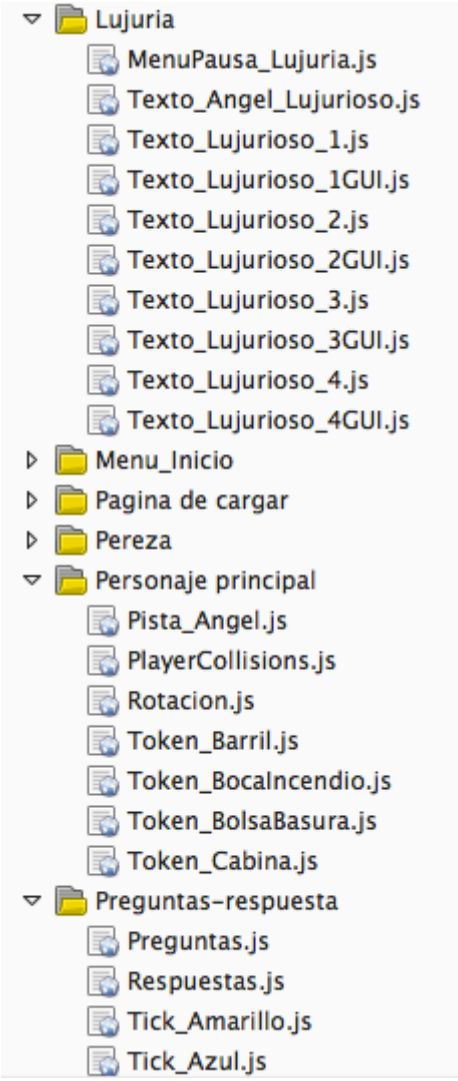


Fig. 4.4. Árbol de carpetas y archivos relacionados con la Lujuria, y la escena de las preguntas.

Como se pudo observar en la figura 4.3, y con respecto a la imagen 4.4, la cantidad de archivos incluidos en esta última es significativamente mayor, esto es debido a que son necesarios para poder trabajar de una manera más sencilla con el desarrollo de la aventura.

En primer lugar se creó un sistema de funcionamiento del personaje principal, individualmente al que Unity ofrece por defecto, de tal manera que la programación incluida en sus archivos, realizasen las acciones necesarias para poder continuar con la aventura:

- Tokens. Inicialmente fue necesario crear los objetos que señalizaran cuando los diálogos se habían acabado, y los cuales marcarán si se puede hablar con el ángel o no, dando pistas en tal caso.
- Rotación. Cada vez que un token aparece en escena, se le ha otorgado la propiedad de girar, hasta que el usuario lo coge.
- PlayerCollisions. Es la parte central de esta escena, ya que en él se incluyen las interacciones que se van a producir, y en caso de que algunas de las condiciones establecidas no se cumpla, se llame a los GUIText (Ver apéndice B), que contendrán frases específicas de cada caso.
- La Pista_Angel. Es el encargado de ofrecer los distintos mensajes al usuario en caso de no cumplir las especificaciones, o en caso contrario poder realizar la pregunta.

Una vez creados los programas principales relacionados con el usuario y sus acciones, es hora de especificar la escena de la Lujuria desde un punto más técnico dentro de su diseño:

- Cada uno de los personajes dentro de la escena, cuenta con dos archivos adjuntos:
    - Texto_Lujurioso_xGUI. A este primer programa, se le ha dotado la capacidad de poder indicar en cada caso que ha de hacer el usuario cada vez que se acerca a los PNJ's. De este modo, en caso de querer hablar con la lujuriosa número 3, se le indicará que es necesario primero un objeto y posteriormente otro, todo ello apareciendo por pantalla con los GUIText, la herramienta incluida dentro del programa Unity.
    - Texto_Lujurioso_x. Este programa es el encargado de poder ejecutar las conversaciones múltiples, y dar la recompensa al usuario en caso de terminar satisfactoriamente.
- Texto_Ángel_Lujurioso. Como en las demás escenas, ofrece las preguntas al usuario, con la particularidad de que en este caso, y gracias a los objetos recogidos, se le indica con algunas pistas sencillas, cuales son las respuestas correctas.

Una vez se postula al usuario la pregunta del ángel, se han dispuesto dos posibilidades: que acierte, o en un caso bastante improbable, que falle. La primera de las opciones, conduce al jugador a la siguiente escena, pero en caso de fallar, se envía al usuario a una escena nueva, la cual no puede ser visitada desde otro lugar sino es este, en el cual se hacen tres preguntas, a las que hay que emparejar con sus respectivas respuestas.

Fig. 4.5. Escena: Preguntas - Respuetas.

La mecánica de este juego es sencilla. Se han de unir los números con las correspondientes letras. Como se muestra en la figura 4.5, las conexiones deberían ser: 1-A, 2-B Y 3-C. Los scripts usados en esta escena son:

- Preguntas. En este script se guardan las preguntas y las respectivas respuestas. Así mismo en este programa se guardan las posiciones de los bocadillos y los botones para poder probar cada una de las respuestas pulsadas. En caso de fallar al responder, los "*ticks*" hasta el momento acertados desaparecerán y será obligatorio volver a empezar desde cero.
- Respuestas. Imprime por pantalla los resultados de las respuestas correctas y se ejecuta cuando las respuestas son correctas. En caso contrario, al igual que en el script de las preguntas, desaparece todo.
- Tick_Azul, Amarillo o Verde. Este tipo de script, funciona del mismo modo que los tokens en la Lujuria. Cada vez que se responde bien una pregunta, aparare ✓ con la respuesta correcta en la parte inferior de la pantalla.

Cabe recalcar, que todo lo anteriormente citado es una breve introducción a que programas hacen que cosas. Para poder tener un mayor entendimiento a como funcionan dichos programas, la gran parte de ellos se detallan con mayor profundidad en el capítulo: *4.2.3. Estructura de programación*.

4.2.2. Interfaz de usuario.

Inicialmente se determinó que los diálogos deberían ser lineales, sin dar a los usuarios la habilidad de poder interactuar de una manera más activa con el juego. Pero en siguientes versiones se planteó la posibilidad de que los jugadores "hablasen" con los personajes dentro de las escenas. Dicha característica, está creada en un entorno cerrado, esto es, que las respuestas están prefijadas de manera que los usuarios tienen varias posibilidades dentro de un abanico de las mismas, pero que no pueden introducir ninguna frase creada por ellos.

Por tanto en el juego se pueden diferenciar dos tipos distintos de conversaciones: uno simple y otro múltiple.

- **Conversación simple**. Este tipo de conversación es la que se encuentra en la mayoría de las escenas del juego.

Fig. 4.6. Estructura de diálogo simple.

Cada vez que el usuario se acerca a un PNJ, se activa el código de conversación. Una vez que este evento se ha activado, el usuario debe presionar la tecla que se le indica, para poder continuar con la conversación. Estas conversaciones son sencillas, y en ellas se cuentan lo más básico de los temas de cada escena, sin cansar al usuario. En este estilo de

conversación, el diálogo se acaba cuando se pulsa la tecla hasta el último de los diálogos guardados dentro del programa.

- **Conversación múltiple**. En este caso la conversación es lineal, pero se ofrecen alternativas al usuario, la denominada respuesta múltiple.

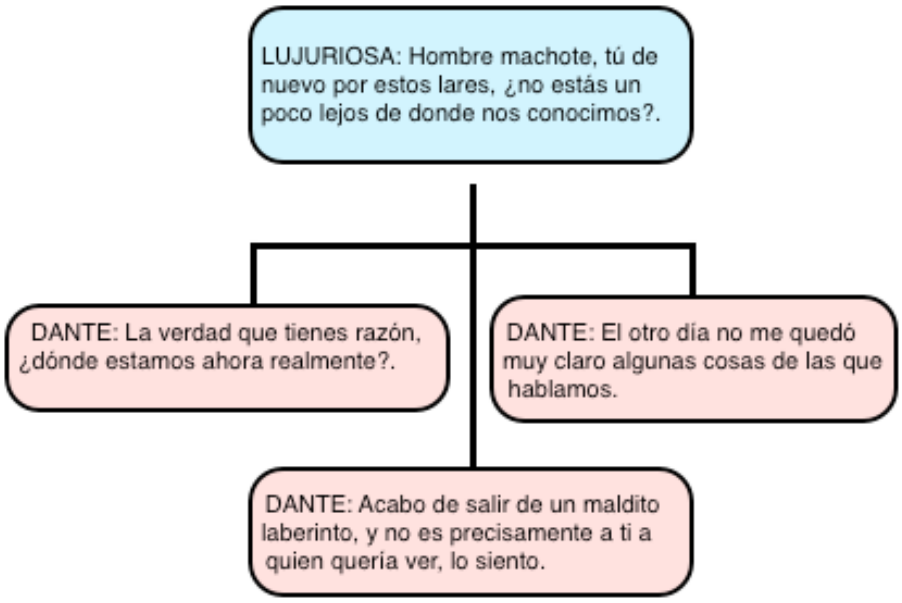


Fig. 4.7. Ejemplo de diálogo con respuesta múltiple.

Este estilo de conversación, es el que se ha creado para la escena de la Lujuria. Como en el caso de las conversaciones simples, es necesario acercarse a cualquiera de los personajes de la escena para iniciar el diálogo. La particularidad, es que el usuario decide cuando terminar las conversaciones o, por el contrario, hacerlas repetir para poder comprender lo que cada uno le ha de explicar. Para poder hacerse una mejor idea de como funciona el orden de conversación dentro de esta escena, se incluye un mapa de diálogo correspondiente a cada uno de los personajes dentro de la escena:

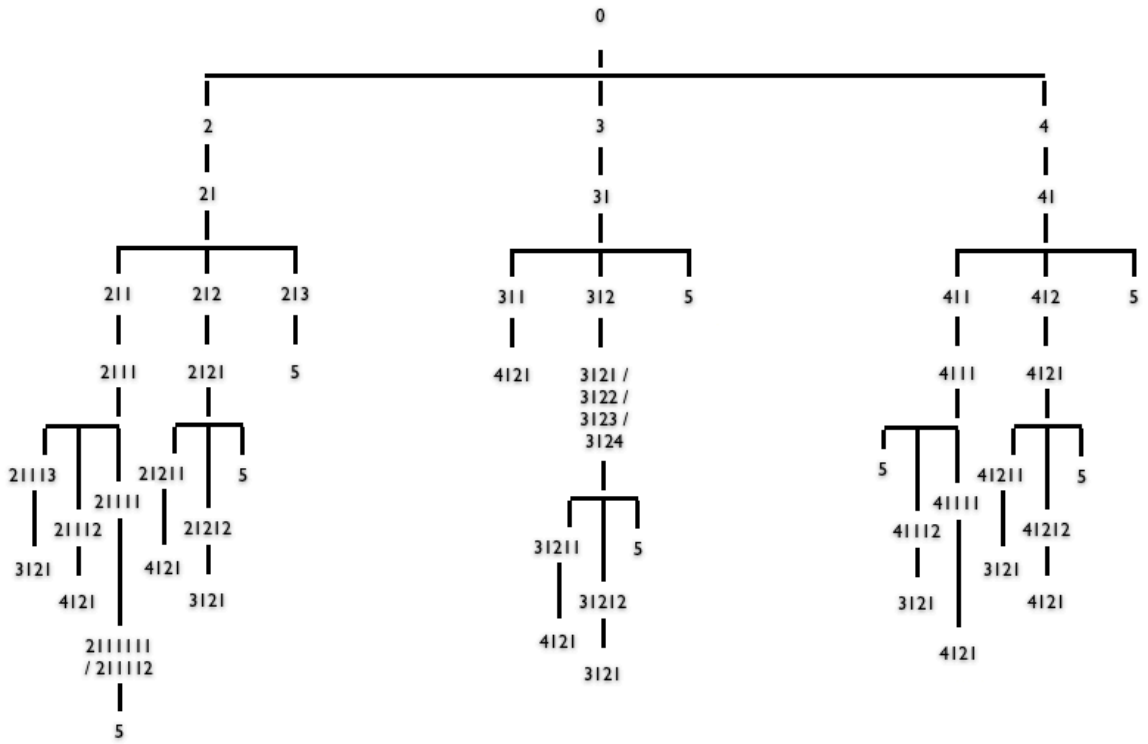


Fig.4.8. Diagrama árbol del diálogo utilizado con *Lujurioso_2*.

Se puede observar que dependiendo de la respuesta que se da, se encamina al usuario a una rama u otra dentro de la conversación, de manera que si en algún momento desea terminar la conversación, solo ha de pulsar la frase correcta.

Otro estilo de conversación que existe dentro del juego, es el creado para los ángeles, pero no se puede considerar conversación, ya que en este caso, sólo ofrecen a los usuarios un examen, en el que han de responder la pregunta. Por lo tanto, no es una conversación en donde se hayan de dar varias respuestas para llegar a un camino final.

En un primer lugar se va a explicar como funciona el flujo de información de las interacciones entre personajes.

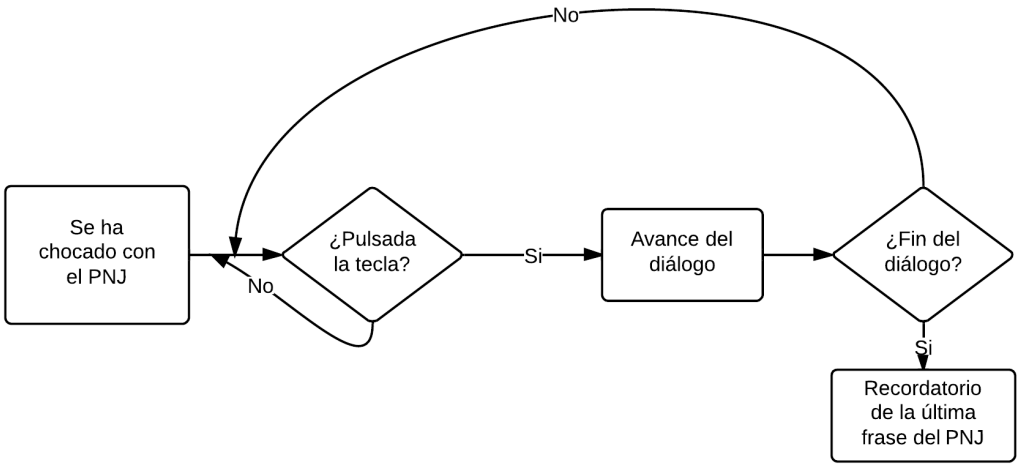


Fig. 4.9. Diagrama de flujo de conversación con cualquier PNJ.

Lo que se representa en esta figura, es la interacción que se produce cuando el jugador colisiona con algún personaje, exceptuando a los ángeles y a los personajes de la Lujuria (ambos se verán como funcionan posteriormente).

Cada personaje del juego tiene creada la misma implementación. Esto implica que todos ellos actúan de la misma forma, siendo esto un problema a la hora de crear varios personajes dentro de la misma escena. Este problema se soluciona con la creación de un sistema de "zona de interacción", en la que sólo se activan los diálogos si se encuentra en dicha zona.

Antes de poder crear cualquier tipo de implementación dentro del código, es necesario definir esa zona de interacción con algún elemento creado dentro de Unity.

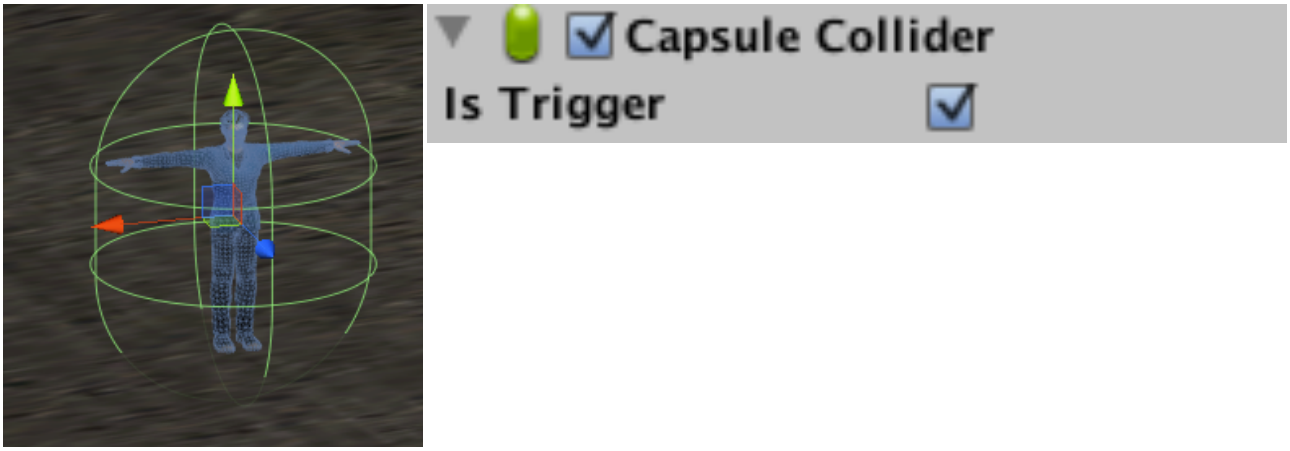


Fig. 4.10. Creación del CapsuleCollider, e incluirlo a un personaje de la escena.

Por tanto es necesario dotar a los personajes del atributo "*Collider*". Esto es la denominada zona de interacción de los personajes. Es necesario que para poder hacer uso de ella, dentro de los atributos, esté activa la casilla IsTrigger, lo cual dice al programa cuando ha de actuar.

Añadiendo el atributo CapsuleCollider a los personajes, no hace nada por si sólo, únicamente indica que se ha chocado con el personaje. Para tal fin, se ha creado un atributo IsEnter, dentro del programa asociado a cada PNJ, el cual da la posibilidad de conectar al personaje con el código asociado a la actividad a realizar, en este caso se activa la conversación.

Para poder hacer avanzar la conversación, es necesario pulsar las teclas del teclado, de tal manera, que es el propio usuario el encargado de avanzar la conversación según la velocidad que le convenga.

Una vez terminada la conversación con cualquiera de los personajes, se puede entrar en otro diálogo con otro PNJ, con los pasos que se han explicado anteriormente, o intentar realizar la prueba del ángel.

El siguiente tipo de conversación implementada dentro del juego, es la prueba que se ha de realizar con los ángeles. Este caso es igual para todos los ángeles de las escenas, de tal forma que el esquema que se presenta, excepto con un adicional diferente que existe con respecto al *Angel_Lujuriso*, es resto es idéntico.

Si las conversaciones se han acabado, o bien el jugador desea realizar la prueba, puede hacerlo en cualquier momento. Esto implica que no es necesario hablar con todos los personajes (se recuerda

que esto no es posible en la escena de la Lujuria), y que se puede intentar realizar la prueba en cualquier momento.

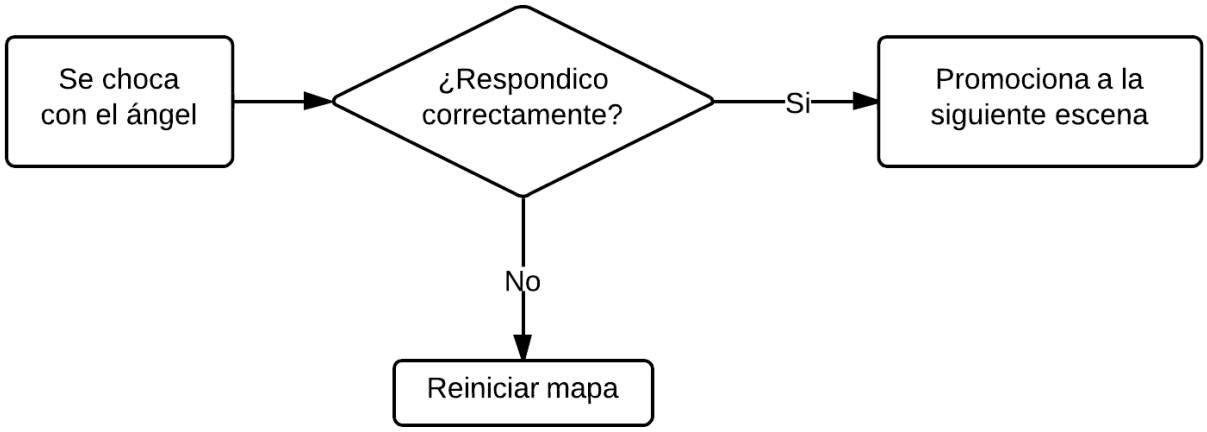


Fig. 4.11. Diagrama de flujo de las conversaciones con el ángel.

El inicio de cualquier conversación se conserva también en este caso, con los BoxCollider activados. En el caso de los ángeles es necesario almacenar las preguntas y las correspondientes respuestas.

Al igual que en el caso de las conversaciones de los personajes, las preguntas se guardan en un array de posiciones, en donde cada posición es el equivalente a una pregunta. Por ello mismo, se crea otro array en donde se guardan las respuestas en la misma posición que las preguntas anteriores.

```
static var preguntas= new Array (); static var respuestas= new Array();
```

Fig. 4.12. Declaración de las variables: preguntas y respuestas.

Una vez seleccionada la pregunta, se seleccionan las 3 posibles respuestas, y se muestran por pantalla en los GUIButton (Ver apéndice B).

La forma de elección de la pregunta es totalmente aleatoria. Entre las respuestas, siempre está la correcta, el usuario ha de seleccionar una de ellas. Si es correcta se accede a la siguiente escena, sino se ha de repetir la escena.

Hasta este punto se han visto los diálogos introducidos en las escenas generales. Como se ha estado indicando, ahora se profundizará en la escena de la Lujuria, donde los textos, varían con respecto al resto de los expuestos.

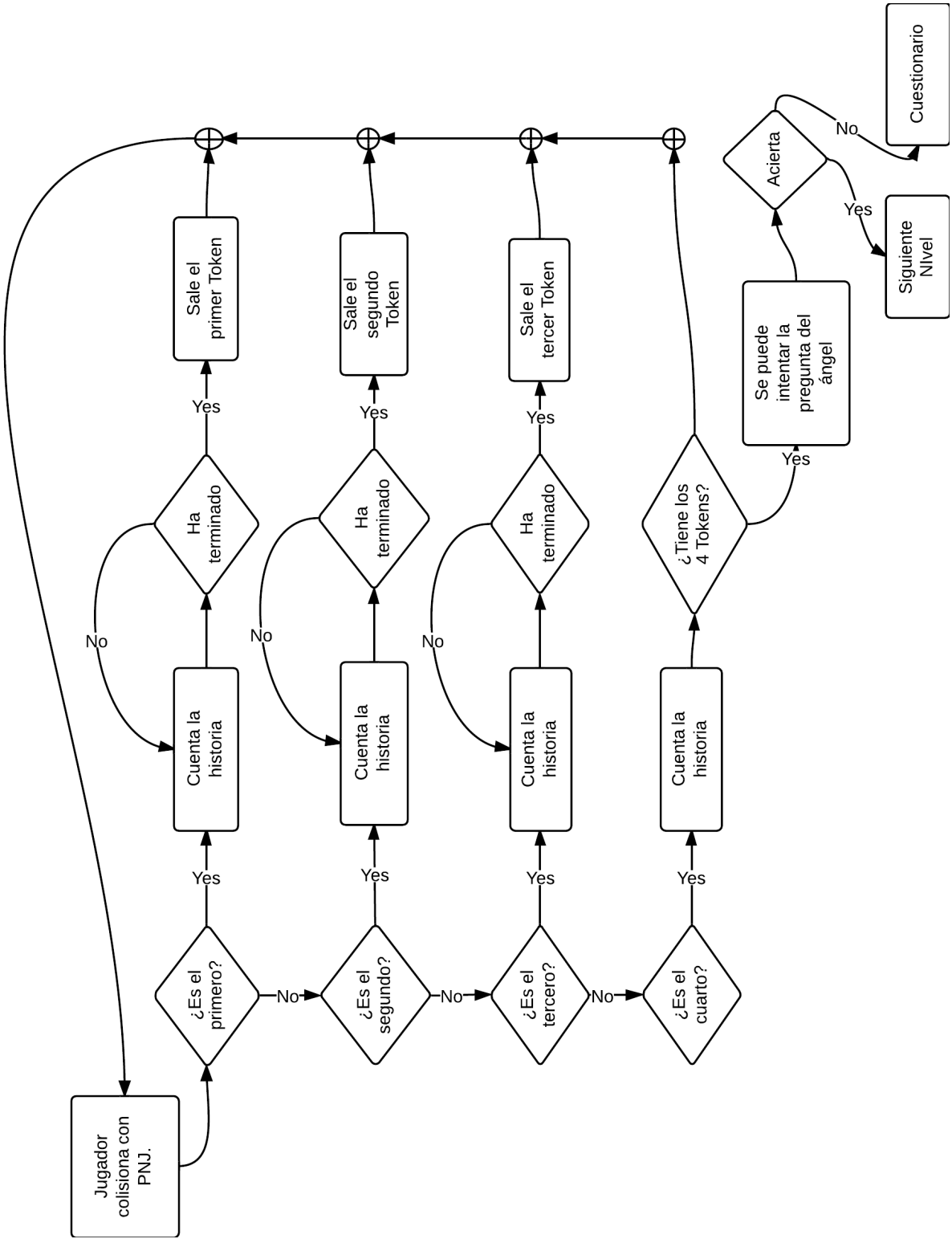


Fig. 4.13. Se puede observar como queda distribuido el diálogo con los personajes.

La figura muestra cual es el funcionamiento de los diálogos en la escena citada. En todas la escena se le ofrece la oportunidad al usuario poder hacer frente a la pregunta del ángel, pero en este caso no es posible tal posibilidad.

Cuando el usuario se acerca al ángel nada más empezar la escena, se encuentra con este mensaje:

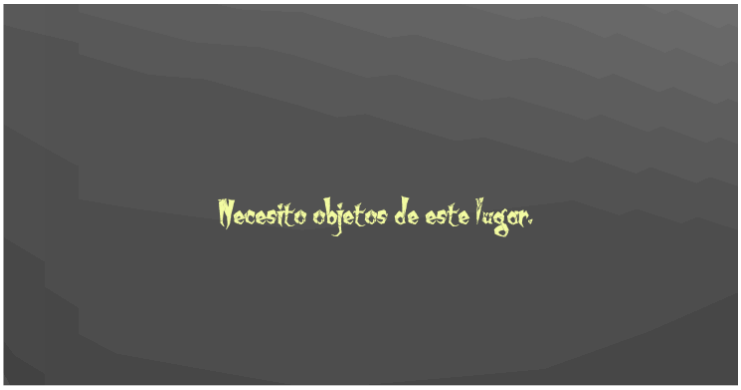


Fig. 4.14. Frase del ángel cuando no se tienen todos los objetos.

Con esta frase, se le indica al jugador que es necesario recolectar objetos de la escena para poder acceder a la pregunta. Dichos objetos serán dados por los PNJ una vez se terminen las conversaciones con ellos.

Los diálogos que se han creado para esta escena son de tipo múltiple (véase el principio de este apartado). El orden de los jugadores está determinado por las explicaciones sobre cada tema que dan, de tal manera que el primer personaje habla sobre las pilas, y el segundo profundiza con los métodos específicos sobre dicho tema. De esta forma es más sencillo asimilar los datos aportados por los personajes, sin tener que confundir términos similares durante su exposición.

De la misma forma que con el ángel no se puede hablar hasta disponer de los objetos pertinentes, también se ha dotado de esta habilidad a los personajes de la escena. No se puede hablar con el personaje que cuenta los métodos de las colas, sin antes saber lo que son las colas.

El usuario verá los siguientes mensajes en caso de no querer seguir el protocolo correcto:

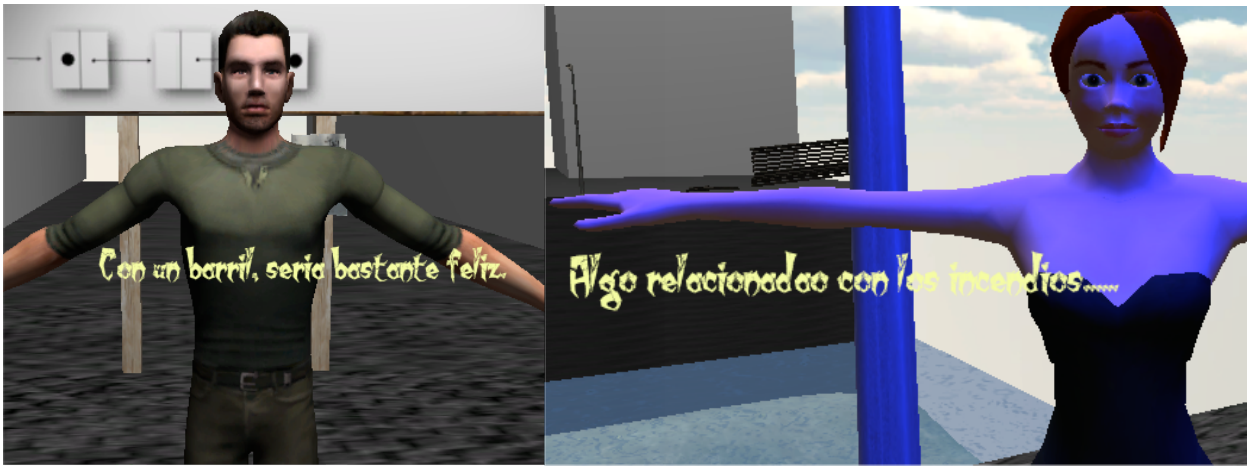


Fig. 4.15. Distintos mensajes de los personajes de la escena Lujuria.

Los mensajes mostrados en la anterior figura, son una muestra de algunos de los que se han incluido en el código de los mismos, ya que si se quiere hablar con el personaje 4 nada más empezar, indicará que necesita un barril, en el siguiente caso, que necesita una boca de incendio, y para terminar una bolsa de basura.

Cuando se habla con el personaje adecuado, se le muestran al usuario las distintas posibilidades a elegir.

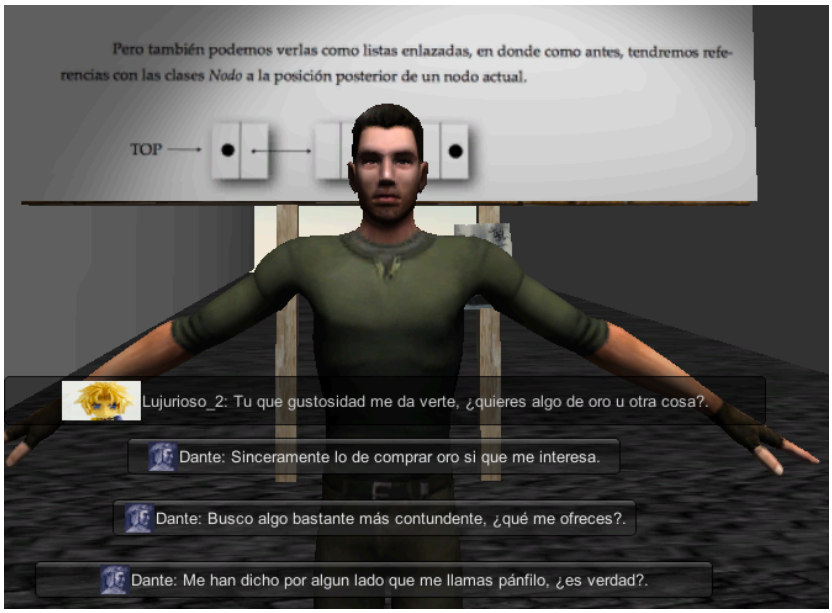


Fig. 4.16. Conversación con uno de los personajes.

Cada PNJ, trata un tema determinado: Pilas, métodos específicos de las pilas, colas y los métodos específicos de las colas. Pero a parte también se les ha introducido otros temas, a modo de hacer más amena la lectura de los temas.

De acuerdo en lo mostrado en la figura 4.16, los diálogos tienen distintos caminos que pueden recorrerse varias veces, dependiendo de si el usuario así lo desea. A cada personaje se le ha dotado de una estructura de flujo similar, pero en ningún caso igual, de manera que el orden seguido para poder terminar una conversación en ningún caso es el mismo.

Cuando el usuario acaba satisfactoriamente la conversación con cualquiera de los personajes, estos dan el token pertinente. Estos tokens son objetos característicos de la escena en la que se encuentra ambientado el pecado.

Fig. 4.17. Representación de los distintos tokens que se les da a los jugadores.

La recolección de los mismos es obligatoria para todos los jugadores, ya que sin ellos no es posible acceder a la pregunta del ángel.

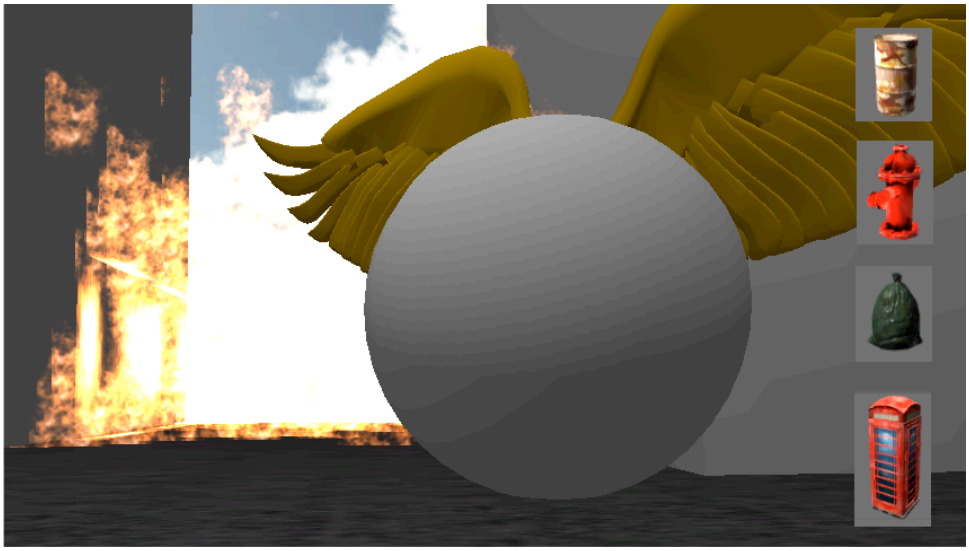


Fig. 4.18. El ángel espera a poner a prueba los conocimientos de los usuarios.

Una vez que se tienen todos los tokens, el usuario puede intentar realizar la pregunta del ángel. Para poder facilitar en cierta medida que la respuesta no sea fallada, cuando se colisione con él, por cada objeto recogido se regala una pista de los temas que han sido tratados.

Fig. 4.19. Pregunta realizada por el ángel de la Lujuria.

Para finalizar este apartado sobre la interfaz de usuario, indicar que en todas las escenas existe la posibilidad de parar el juego, pulsando en cualquier momento la tecla "*Esc*". Si el usuario pulsa la tecla, el juego se para y ofrece ciertas posibilidades de interacción con el usuario, de tal manera que se le permite decidir sobre la acción que desea tomar sobre el juego.

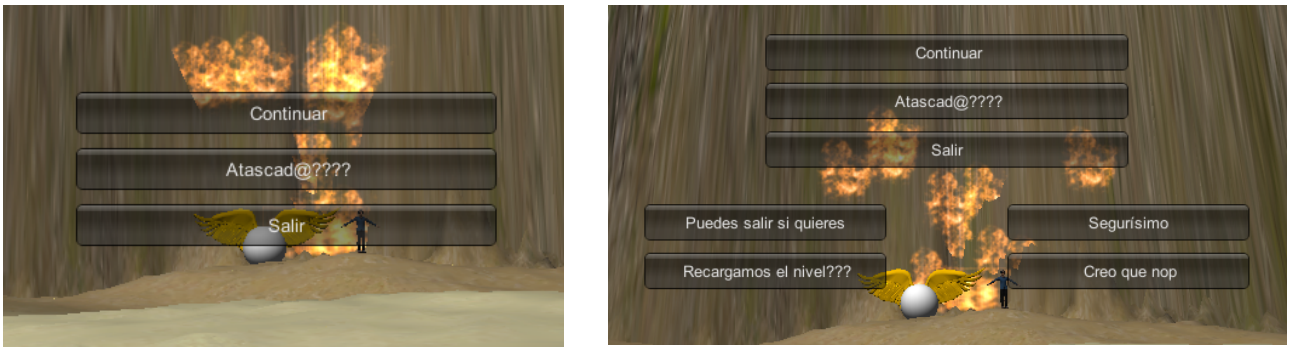


Fig. 4.20. Menú pausa emergente al pulsar la tecla "Esc", y las distintas opciones.

Estudiando las dos fotos que se muestran en la figura 4.20, se observan las distintas posibilidades que se le ofrecen al jugador a la hora de decidir que quiere hacer mientras se encuentre en el menú de pausa.

- Si el jugador pulsa el globo "Atascad@???", se produce una nueva aparición de dos globos:
  - "Puedes salir si quieres".
  - "Recargamos el nivel???".

El primero de los globos ofrece al jugador una manera más práctica de salir de la escena actual y poder salir al inicio del juego, o en el caso del segundo globo, y sólo siendo útil en caso de que el personaje se quede trabado en algún punto sin poder moverse debido a la geografía del terreno.

- O en el caso de pulsar directamente la opción "Salir", se dan las siguientes nuevas opciones:
  - "Segurísimo".
  - "Creo que nop".

  Como en el caso anterior, si se pulsa el primer globo, es una afirmación indirecta para salir de la escena de manera inmediata, sin tener que pasar por otro globo, mientras que en el caso del segundo globo se le da la posibilidad de volver al juego de una manera sencilla y sin tener que pulsar la tecla "*Esc*" una vez más.

Por último, se puede encontrar más información en cuanto a la creación de los bocadillos que contienen la información en el Apéndice B.

4.2.3. Estructuras de programación.

Una vez diseñado la forma en que los scripts van a trabajar, y viendo como van a ser los diálogos que se muestran por pantalla al usuario, es momento de hacer una profundización en cuanto a programación respecta.

En este apartado se mostrarán los distintos flujos de información que se dan lugar durante la ejecución del juego, así como una breve introducción a cada uno de ellos.

Como se ha demostrado en el apartado *4.2.2. Interfaz de usuario,* los personajes interactúan con el protagonista, de tal forma que cada vez que se entra en su zona de interacción el código se activa y empieza a trabajar.

Es necesario dotar a los PNJ´s del atributo "*BoxCollider*", el cual como se pudo observar en la figura 4.10, crea una zona en la cual se activa el código si se entra en ella. Para poder activar dicha función es necesario crearse una variable dentro del código del personaje, el cual indicará en que momento se está en contacto con esta zona.

```
var IsEnter: boolean = false;
```

Fig. 4.21. Atributo que indica si se puede interaccionar con el PNJ.

Una vez activa esta variable, las conversaciones se ejecutan de forma lineal. Los diálogos que se han creado son un conjunto de arrays, en donde cada posición es una frase.

```
public var Dialogs: int = 0; //Para decir qué texto mostrar.
private var iDialogueCnt:int = 0;
private var aDialogs= new Array();
```

Fig. 4.22. Variables que guardan el estado de las conversaciones.

Una vez que ya se ha empezado la conversación con cualquiera de los personajes, es necesaria la intervención del usuario con el teclado. Al principio de cada diálogo se indica al jugador que tecla ha de pulsar para poder hacer avanzar la conversación. Esto es debido a que cada PNJ, tiene asociada una letra distinta.

Con este planteamiento, se tuvo un inconveniente, si se pulsaba una misma tecla para todos los personajes, y se hablaba con cualquiera de ellos, el otro también sufría un avance en sus diálogos. Por lo que finalmente se optó por dotar a cada personaje de una letra distinta.

```
//Dependiendo de la tecla que se pulse nos saldra una condicional u otra
if(Input.GetKeyDown(KeyCode.R) && Dialogs < iDialogueCnt)
{
Dialogs += 1;//Dialogo Adelante.
}
if(Input.GetKeyDown(KeyCode.T) || Input.GetKeyDown(KeyCode.E)){
Dialogs = 0;
}
```

Fig. 4.23. Ejemplo del pulsado de teclas para avanzar en la conversación.

Si se pulsa la tecla asociada al personaje correspondiente, el contador *Dialogs* aumenta en una unidad. A partir de este punto cabe destacar las dos posibilidades que se han dotado a este sistema:

- La primera es que si el usuario sigue pulsando la tecla correcta, el diálogo avanza correctamente, hasta llegar al final del mismo, que dejará impreso por pantalla el último mensaje que se ha dicho, y que sólo quedará activo si no se ha habla con otro personaje.
- La otra que se ha añadido, es la posibilidad de iniciar el sistema de diálogos durante cualquier conversación. Mientras la conversación esté activa, se ha de pulsar la misma tecla, en caso de pulsar otra, el diálogo empezará desde cero, como si fuese la primera vez que se habla con el personaje.

Esto se puede ilustrar con un ejemplo detallado en varios pasos:

1. Se colisiona con el primer personaje, suponiendo que la tecla asociada a dicho personaje es la “E”, se presiona tantas veces como sea necesario para terminar el diálogo.
2. Si durante el proceso se pulsa la tecla “R”, la cual está asociada al segundo personaje, el contador de diálogos se inicia a cero, indicando nuevamente cual es la tecla que se ha de pulsar para poder continuar.
3. Una vez finalizado el diálogo, al iniciar uno distinto con el segundo personaje, si se pulsa la tecla “R”, el contador de diálogos se encuentra inicializado a cero, por lo que la conversación puede empezar desde el primer diálogo.

Al terminar la conversación con el personaje pertinente, y a modo de recordatorio, se ha programado la posibilidad de aparecer la última frase del diálogo únicamente si no se pulsa otra tecla, y sólo aparece si se entra en contacto con el personaje en cuestión.

Las conversaciones están todas programadas de la misma manera que se ha explicado anteriormente. Pero la particularidad de la Lujuria radica en que es necesario disponer de los tokens especiales para poder empezar a conversar con cualquier personaje de esa escena.

Cada token de la escena Lujuria tiene asociado un valor, el cual sólo se activa cuando el personaje principal dispone de él.

```
static var tokens : int = 0;
var token : Texture2D;
```

Fig. 4.24. Atributos asociados a los tokens.

De esta manera, para poder activar la pestaña correspondiente a la posesión del Barril (es el primero de los objetos que se dan), el programa cambia en el código *tokens = 0*, por *tokens = 1*. Así el sistema sabe que el último token que se ha cogido, es aquel cuya referencia es igual al valor que se haya otorgado durante la ejecución del programa.

Las conversaciones que se tienen durante el transcurso de la escena, permiten acceder al valor de los tokens, haciendo que varíe con respecto se va avanzando en la historia.

Durante las conversaciones en la Lujuria, la variable IsEnter también está presente, pero se le añade un adicional para que la función sepa en que casos se está activando.

```
Texto_Lujurioso_2.lujurioso_2 = true;
Texto_Lujurioso_2.isEnter = true;
```

Fig. 4.25. Activación de los diálogos.

Para poder empezar la conversación es necesario disponer del token apropiado. Cuando este requisito se cumple, y se hable con el personaje correspondiente, los diálogos aparecerán por pantalla, y el usuario podrá pulsar el que crea mas oportuno dependiendo de la información que desee conocer. Todo esto se controla mediante un GUIBox, que muestra por pantalla las frase seleccionadas y los GUIButton, para seleccionar las opciones pertinentes. (Ver Apéndice B)

Este proceso se repite hasta que el jugador decida dar por zanjado el tema, y pulse la frase que cierra el diálogo. Una vez pulsada esta opción se ejecutan distintas partes del código. En todos los personajes se creó la variable antes citada, IsEnter, pero por un error de programación, esta no se cancelaba cuando se salía de la conversación, sino que quedaba activa, lo que provocaba que a la hora de intentar hablar con otro personaje el tiempo del juego continuase y se pudiesen montar los distintos diálogos sin saber realmente que frase se había pulsado. Por esa misma razón una vez terminada la conversación, se asigna un valor de *false* a dicha variable, para que no vuelva a suceder el citado error.

Una vez que el objeto es presentado ante el usuario, este girará, gracias a la función añadida y anteriormente citada *Rotacion.js*. Mientras el usuario no coja el token, cada vez que se acerque al último personaje, indicará con una frase que aún le queda algo por hacer con él/ella.

El objeto que aparece, en un principio era infinito, es decir aparecían infinitas copias del mismo, de tal manera que provocaba un colapso en el sistema a la hora de continuar con la ejecución del mismo. Se decide en tal caso, que una vez que se entrase en dicho bucle, solo se pudiese hacer una única vez, anulando la iteración internamente con la variable *repetición*.

```
if(lujurioso_1==true){

    if(repeticion==false){

        lujurioso_1=false;
        Instantiate(barril, Vector3(52, 12, 23), transform.rotation);
        repeticion=true;
```

Fig. 4.26. Anulación interna de la aparición de varios tokens.

Una vez que se van consiguiendo ordenadamente los tokens ofrecidos, los usuarios deben encaminarse a la prueba del ángel. En este caso, también se le ha obligado a tener que saber si el jugador dispone de los tokens necesarios para poder hacer la prueba, ya que en caso negativo únicamente aparecerá el mensaje ilustrado en la figura 4.14.

La prueba de este ángel, es igual que las del resto de escenas, con la particularidad de las pistas que dan los tokens recogidos. Pero aún así, se ha decidido poner un pequeño castigo a todo aquel que falle cualquiera de las preguntas que se le plantea.

La escena creada para tal fin, es un simple juego en que se han de unir preguntas con respuestas. En este caso el uso de las GUI, ha sido la parte fundamental. Como en el caso de los ángeles, se dispones de una batería de preguntas, con sus respectivas respuestas guardadas ambas en un array.

```
static var preguntas= new Array ();
static var respuestas = new Array();
```

Fig. 4.27. Declaración de variables.

Cada pregunta aparece incluida dentro de un GUIBox, y a su lado un valor numérico o una letra. Para saber si se ha contestado bien, el usuario verá aparecer en pantalla un ✓ con la correspondiente respuesta correcta. Si se intenta superar esta escena contestando tres veces la misma pregunta correctamente, se ha provisto de una función, donde cada vez que se intenta hacer dicha trampa, indica que ya se ha respondido y que ha de intentarlo con otras posibilidades. Y para concluir, en caso de equivocarse en alguna de las respuestas, se hace un reseteo de todo lo anterior, y el usuario ha de empezar a relacionar todo nuevamente.

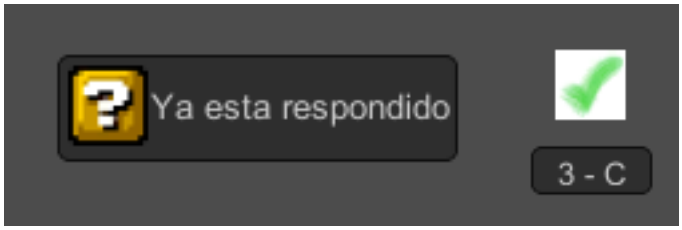


Fig. 4.28. Cuando se repite alguna contestación, y como se muestra la respuesta correcta.

En el siguiente apartado se puede ver, en mayor profundidad como funciona el código mas explícitamente.

4.3. Implementación del código.

Hasta el momento, se ha desarrollado por encima, en algunos casos con un poco más de especificación, el trabajo que realizan ciertas funciones y las distintas partes que componen la gran mayoría del código.

En este nuevo apartado, se van a ilustrar las funciones más representativas, asociadas a las funciones importantes durante la ejecución del programa.

Como se ha podido demostrar a través de los anteriores capítulos, la escena de la Lujuria es el eje principal en el que giran el resto de las escenas, ya que es en ella en la que se ha hecho más hincapié sobre la programación usada y la interfaz usuario.

Por ese mismo motivo, en este apartado, se va a profundizar en lo más representativo de la escena, pues contiene en gran parte la esencia del resto de las escenas.

Como se ha detallado en el anterior apartado, todas las escenas tienen diálogos de tipo lineal, excepto en la Lujuria, donde se pueden encontrar diálogos de respuesta múltiple.

En el resto de las escenas, se dejó aparcada la programación de los actos del personaje principal, derivándolos a un acto único lanzado por cada uno de los PNJ. En este caso, se creó un programa único asociado al personaje, de tal manera que la interacción fuese principalmente del personaje controlado, y no como en el resto de ocasiones, que se impone por los personajes distribuidos en la escena.

Para poder interactuar con cualquiera de los personajes que se han ubicado en la escena, es necesario habilitar dicha posibilidad con el atributo IsEnter:

```
Texto_Lujurioso_2.lujurioso_2 = true;
Texto_Lujurioso_2.isEnter = true;
```

Fig. 4.29. Activación del diálogo con el personaje Lujurioso_2.

En caso de no tener el objeto relacionado al personaje en cuestión, se ha habilitado la función de no permitir empezar el diálogo hasta no disponer de ella:

```
Texto_Lujurioso_2GUI.textOn = true;
Texto_Lujurioso_2GUI.mensaje = "Con un barril, seria bastante feliz.";
```

Fig. 4.30. En caso de no disponer del objeto en cuestión.

El ejemplo mostrado es el referente al personaje *Lujurioso_2*, lo que lleva a la pregunta ¿por qué no se muestra el *Lujuriosa_1*, ya que es el primero? La respuesta es sencilla. En el caso de *Lujuriosa_1*, no es necesario disponer de ningún tipo de token para poder hablar con ella, pero en el caso del segundo personaje, es necesario disponer de un objeto, y también es el primero que indica que se necesitan objetos para poder continuar con los demás diálogos.

Continuando con lo expuesto en la última figura 4.29, una vez que se ha chocado con el espacio personal del personaje, y se dispone del correspondiente token, el diálogo empieza. Los tokens que

se han ideado son 4, y no están repartidos a la vista del jugador, sino que estos aparecen según se va hablando con los distintos personajes.

Fig. 4.31. Representación de los distintos tokens que se les da a los jugadores.

Los tokens presentados en la figura, representan elementos de la vida diaria de las personas, acorde a la escena de ciudad que se ha intentado dar a la escena en sí. Para poder interactuar con cada uno de ellos independientemente se realizó una denominación especial a cada uno de ellos en el inspector dentro del programa, ya que de esa manera, sólo se activasen ciertas funciones cuando se consiguiese recolectar cada uno de ellos sin importar que es lo que se hacía con los otros.

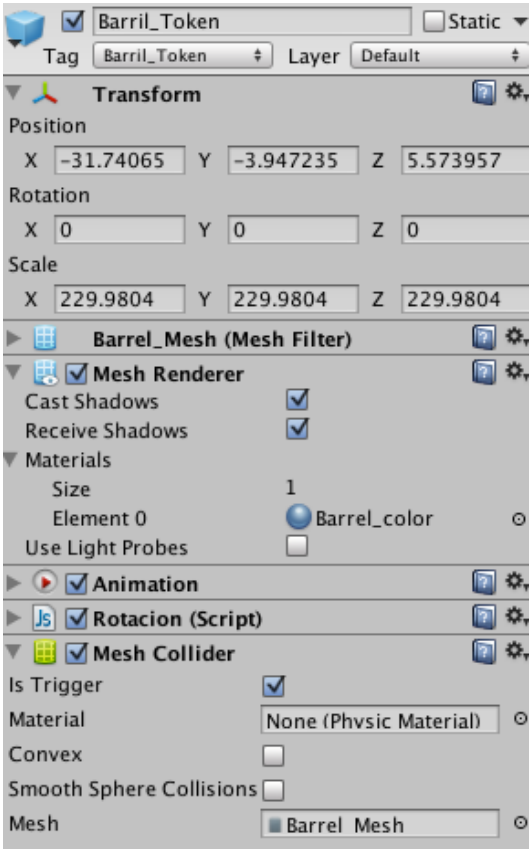


Fig. 4.32. Declaración de los valores necesarios dentro del inspector.

Una vez que se hubiese cogido cada uno de ellos se puede acceder a la siguiente conversación, continuando con esta mecánica hasta poder hablar con el ángel. El IsTrigger ha de estar activo, para que cuando se colisione con ellos, se ejecuten las acciones programadas específicamente para ello.

Para poder hacer que cada uno de los tokens tenga en cierta medida vida propia, se decidió crear un programa específico para cada uno de ellos con esa función específica: *Token_Barril.js, Token_BocaIncendio.js, Token_BolsaBasura.js y Token_Cabina.js*. Cada uno de ellos, asociado a su objeto pertinentemente, es invocado cada vez que se le atraviesa, de esa manera es capaz de aparecer a la derecha de la pantalla.

```
static var tokens : int = 0;
var token : Texture2D;

function Start () {
        guiTexture.enabled = false;
}

function Update () {
    if(tokens == 1){
        //Una vez que se coja el primer token, queda activada su aparicion e la pantalla
        guiTexture.texture = token;
        guiTexture.enabled = true;
    }
}
```

Fig. 4.33. Programas asociados a cada uno de los tokens.

Con lo expuesto se pretende que no aparezcan todos los tokens a la vez, por lo que se hace una limpieza de cada uno de ellos, de tal manera que sean independientes y sólo se activarán en caso de que el jugador principal lo active mediante las conversaciones.

Todas las conversaciones tienen un final común, la casilla dentro del array de diálogos con el valor cinco (Este hecho se puede observar en la figura 4.8). Esta casilla determina que se ha llegado al final del diálogo, y por lo tanto se puede salir de la conversación y recoger el premio. El Atributo determinado para indicar el final de la conversación es: *Dialogs = 5;* .

Una vez terminada la conversación, se ha creado una forma de obligar al usuario a coger el objeto y que no se olvide de él, ya que si no lo recoge, es imposible proseguir en la aventura.

```
if(Dialogs == 5){
    //ULTIMO MENSAJE ANTES DE SALIR DEL MAPA DEL LABERINTO

    Time.timeScale = 1;

    if(Dialogs == 5 && Token_BocaIncendio.tokens != 2){
        //Se repetirá hasta que no se coja el token....
        Texto_Lujurioso_2GUI.textOn = true;
        Texto_Lujurioso_2GUI.mensaje = "Ten cuidado que mi hermano anda escondido...";
    }

    if(lujurioso_2==true){

        if(repeticion==false){

            lujurioso_2=false;
            Instantiate(bocaIncendio, Vector3(27, 12, 134), transform.rotation);
            repeticion=true;
```

Fig. 4.34. Parte del código de finalización del diálogo.

El principal problema que se produjo en esta secuencia, fue que una vez finalizada la conversación, por pantalla todavía se quedaba el último de los mensajes. Por esa razón se optó por un GUI.Text para que se quedase el mensaje en pantalla, sólo lo necesario.

Una vez que aparece el mensaje flotante en pantalla, se actualiza el mensaje para continuar presente mientras no se coja el token. La sentencia de coger el objeto está incluida en un *if*, ¿la razón? La primera vez que se crea el objeto, lo hace infinitas veces, y hasta que se coge, no desaparece. Por eso mismo, se obliga a que entre en la sentencia una vez con el atributo *repetición = false*, ya que únicamente se ve modificado dentro de esa misma sentencia, lo que produce que sólo se entre una vez y una vez se salga del método no vuelve a entrar.

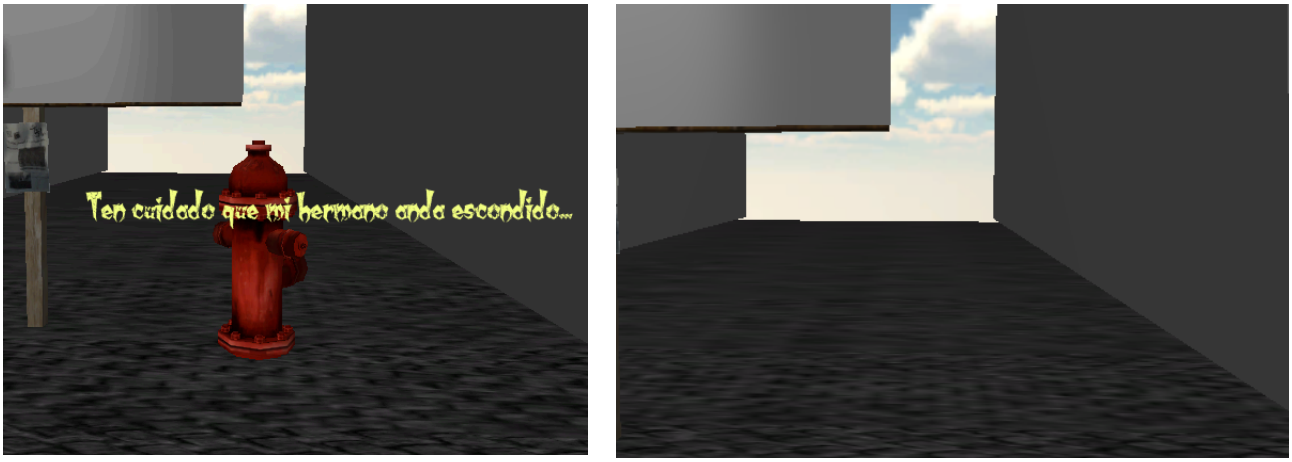


Fig. 4.35. El antes y después de coger el mismo token.

Terminado de explicar el tema de funcionamiento de las conversaciones, toca explicar el funcionamiento del código asociado al ángel de esta escena. La diferencia real es muy poca con respecto el resto de los ángeles, en lo referente a las preguntas, sigue el mismo código, pero como se ha indicado, es necesario recuperar todos los tokens para poder activar dicho diálogo.

```
}else if(hit.collider.gameObject.tag == "Angel" &&  Token_Cabina.tokens == 4 && Token_Barril.tokens==1 &&
    Token_BocaIncendio.tokens == 2 && Token_BolsaBasura.tokens == 3){

    Texto_Angel_Lujurioso.isEnter = true;
```

Fig. 4.36. Código que permite realizar la prueba del ángel.

Una vez que se han conseguido todos los objetos, y se habla con el ángel, aparece en la pantalla de preguntas los tokens asociados a cada uno de los temas que se han explicado durante el desarrollo de la escena.

Fig. 4.37. Pregunta realizada por el ángel de la Lujuria.

Cada uno de los tokens que se han recolectado, dan información detallada y precisa sobre cada una de las explicaciones que han dado los personajes en la escena, de esta manera el primero dará pistas sobre las pilas y las LIFO, el segundo sobre sus métodos específicos, y así sucesivamente como se ha explicado.

Como en el caso de los ángeles anteriores, si se contesta correctamente, se transporta al usuario al interludio entre escenas, pero ¿y si se equivoca? En caso de equivocarse, no se le transporta de nuevo a la escena en la que se encontraba, sino que en este caso se le lleva a una nueva escena en la que tendrá que resolver un pequeño escenario en donde se han de unir preguntas con respuestas.
Esta escena es una penitencia, en la que para poder responder bien las preguntas planteadas, han de haber asimilado la información ofrecida en las anteriores escenas, de modo que no se pueda obviar nada

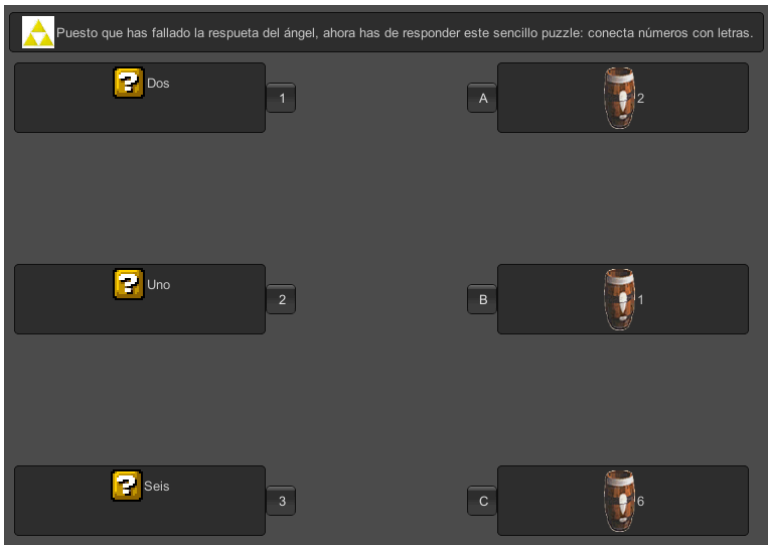


Fig. 4.38. Escena en caso de fallar la pregunta del ángel.

Lo que corresponde a cada un de las cajas, son como en el caso de los diálogos de los personajes de la Lujuria, los métodos usados GUI.Box y de GUI.Botton. En el caso que se muestra en la figura, no se han incluido aún las preguntas y sus respectivas respuestas, por lo que para saber que las preguntas están bien relacionadas, se han sustituido estas por los números para saber su orden.

Es necesario crear un sistema de comparación de preguntas, para no darse el caso en el que todas las preguntas sean iguales, de este modo siempre aparezcan distintas preguntas y sus respectivas respuestas asociadas en un orden que no tenga que ser el mismo.

```
if(preguntaSeleccionada1 == preguntaSeleccionada2){
    dato2 = Random.Range(0, preguntas.length);
```

Fig. 4.39. Forma de comparación de las preguntas seleccionadas.

En caso de que alguna de las preguntas se repita, es decir sea la misma que otra, se ha creado un sistema de comparación para en caso de ocurrir, cambiar la opción de las preguntas.

De igual manera que en el caso de los ángeles, se ha creado una batería de preguntas, de forma que no exista posibilidad de repetirlas. En este programa se han añadido seis posibilidades, pudiéndose introducir tantas como se desee, pero es necesario habilitar las posibilidades dentro del programa.

```
posibilidades = Random.Range(0, 5);
```

Fig. 4.40. Declaración del atributo que permite seleccionar las preguntas.

Cada vez que se contesta correctamente cualquiera de las preguntas, se imprime por pantalla la contestación asociada de tal forma que el usuario pueda ver cual ha sido su elección y si ha sido la buena. Ha sido necesario crear un programa denominado *Respuestas.js*, en el que se han guardado las posibilidades de contestación, las cuales se mostrarán por pantalla.

```
Tick_Verde.tick = 3;
Respuestas.letrero = 8;
```

Fig. 4.41. Definición de la imagen a mostrar y la respuesta correspondiente

Una vez se haya conseguido conectar las tres respuestas, se promocionará al personaje al interludio siguiente, de tal manera que el jugador pueda continuar con la aventura.

```
//ya se ha respondido bien todas las casillas
if(correcto == 3){
    Time.timeScale = 1;
    Tick_Amarillo.tick = 0;
    Tick_Azul.tick = 0;
    Tick_Verde.tick = 0;
    Application.LoadLevel("Lujuria");


}
```

Fig. 4.42. Se carga la escena de la Lujuria y se reinician los contadores.

Otro problema que se tuvo que hacer frente con una de las actualizaciones de los programas, se produjo en que ha sido necesario inicializar al personaje en todas las escenas, ya que si se ejecutaba una escena desde la página de carga, el personaje, únicamente era capaz de girar la cámara sin posibilidad de moverse.

4.4. Conclusiones.

Este capítulo es la espina central del trabajo entero. Se han explicado como están desarrolladas las escenas de los pecados capitales, cual es la forma de funcionamiento de los diálogos frente a la interacción del usuario, para terminar viendo lo más representativo, en cuanto a la programación de las funciones principales respecta.

En todo momento se ha intentado diferenciar la escena de la Lujuria, principal motor de actividades del juego, con respecto al resto de las escenas, de manera que siempre se pudiesen observar claramente las diferencias que existen entre estas y aquella.

Para finalizar, indicar brevemente, que el punto *4.3.Estructuras de programación*, no se ha profundizado en el funcionamiento de los atributos o funciones, ya que para ese tipo de acometido existen los libros dedicados específicamente a la programación sobre JavaScript, o más brevemente con la guía rápida incluida al final de este trabajo a modo de anexo (Ver apéndice B).

Capítulo 5

Batería de pruebas.

5.1. Resultados.

El trabajo que se ha realizado es para el desarrollo de los alumnos en las asignaturas relacionadas con programación, más específicamente con la materia relacionada a la programación en Java.

El proyecto, durante la fase final de desarrollo, se dejó probar a compañeros de la misma carrera, así como a personas ajenas a cualquier asociación con asignaturas de programación. Se les entregó un breve cuestionario, y estas fueron sus respuestas a las preguntas realizadas:

- **Facilidad de uso en el juego.**
  - En casi todos los casos se ha indicado que no supone grandes dificultades de uso la herramienta, ya que los controles básicos del personaje: tanto las teclas WASD como los cursores de dirección sirven para manejar al personaje.

- **¿Ha sido sencilla la asimilación de los temas tratados en las escenas?**
  - En este caso se han dado varias opiniones de distinta índole:
    - Entre las personas que conocían la materia, indican que sí ha sido sencilla la asimilación de los temas y que los ejemplos propuestos de la vida cotidiana, hacen más sencillo la comprensión de lo que se quiere explicar.
    - El otro caso, la gente que no tenía relación con la programación, indica que los términos usados, al ser de una temática específica de programación, en algunos de los casos, no se podía entender lo que significaba, debido a no disponer de una base básica.

- **¿Es necesario tratar algún tema que no está incluido dentro del juego?(Esta pregunta se dirige principalmente a los que han cursado alguna vez este tipo de asignaturas).**
  - Lo que se indica, en términos generales, es que siempre se podría introducir más temas de los ya tratados, pero en palabras de uno de los testadores: *" [...] el juego en sí ya es bastante técnico, y cuantos más temas se introduzcan más largo se haría [...] "*.

- **¿Lo recomendarías a los estudiantes de primer curso como una herramienta de apoyo?**
  - Todos aprueban el uso de la herramienta para alumnos de primero que cursan esta parte de la programación por primera vez, y en palabras de otro de los testadores, en este caso una persona sin contacto con el mundo de la programación: *"Es recomendadísimo a los estudiantes de primero ya que aprenderían de una forma entretenida todo lo necesario para tener las bases de la programación y ayuda a asimilar los conceptos que vayan aprendiendo durante las clases "*.

- **Indicar que cambios harías en cualquier aspecto del juego.**
  - Esta última pregunta sacó a la luz ciertos defectos relacionados con las faltas de ortografía, así como problemas en la programación de diálogos de algunos personajes, como el caso de Obelix en la soberbia, al cual en la sentencia de código se le había incluido una linea más de comentario, sin ser necesaria, lo que provocaba que el jugador se quedase atascado en esa conversación sin posibilidad de pulsar cualquier tecla, obligándole a tener que cerrar el programa y volver a ejecutarlo.
  - Un problema que se presentó y se tuvo que solventar de manera inmediata, fue en el caso de la Lujuria, y la escena de preguntas y respuestas. Una vez superada la prueba de las preguntas, se devolvía al jugador a la escena de la Lujuria, pero con el inconveniente, de que nada más empezar la escena, se presentaba por pantalla las

posibles preguntas del ángel, de manera que el usuario no podía hacer nada excepto contestar las preguntas e intentar promocionar a la siguiente escena.

Generalmente todos los que lo han probado aprueban su uso como un apoyo general a la asignatura, indicando además cuáles, desde su punto de vista, serían las posibles mejoras para la actualización 2.0 (posible multijugador, sistema más complejo en cuanto acertijos-recompensas, posibilidad de realizar pequeñas prácticas en una consola virtual para probar lo aprendido o un sistema de guardado más complejo).

Capítulo 6

Trabajos futuros

Las facilidades que proporciona Unity para la creación de juegos, son sencillas para cualquier estilo, desde básicos juegos de golf, hasta un Shooter, para deleite de los usuarios, pero sobre todo para deleite de los creadores de juegos no profesionales.

Fig. 5.1. Shooter creado íntegramente en Unity 3D.

La posibilidad de creación de juegos en Unity, no sólo se basa en juegos de género violento, o de acción. También se pueden observar los distintos tipos de juegos citados en el capítulo 2: carreras de coches, deportivos, aventura gráfica, puzzles, o lo que aquí se ha tratado, juegos educativos.

6.1. Mejoras para una versión 2.0.

El juego completo se ha realizado con una versión básica de Unity, por lo que los acabados, escenarios y posibles visualizaciones de los entornos, no son del todo comerciales.

Una de las principales actualizaciones que se espera poder realizar con la nueva versión, es mejorar el contenido didáctico de la aplicación, con puzzles mas complejos, más personajes y un sistema de recompensas para poder hacer sentir al jugador más cercano al juego.

Poder hacer de este título un juego multijugador es también uno de los principales objetivos que se perseguirían en la actualización. Con esta idea se intentaría poder conectar a los distintos usuarios, para intercambiar opiniones, realizar distintos restos conjuntamente, o simplemente hacer uso del juego para mantenerse en contacto y poder hablar de lo que cada uno quisiera. Con esta idea se empezó a trabajar en el juego, pero la falta de servidores gratuitos, dejó de lado esta idea para centrarse en el jugador como único usuario.

También se puede expandir esta misma idea de juego, no sólo a un entorno de programación dedicado a Java, sino que con la correcta utilización de las escenas, y una ambientación adecuada para cada caso, se pueden ofrecer distintas herramientas para otros ámbitos de estudio, como por ejemplo, el estudio de señales en las asignaturas relacionadas a las redes, o a otro tipos de lenguajes como C++, Pascal y por que no ensamblador.

La introducción de personajes no estáticos es otro avance que se espera poder resolver con el uso de programas ofertados por Unity, así como un menú inicial, para que cada usuario pueda elegir la creación del personaje acorde a sus gustos. El sistema de guardado y de cargado en esta primera versión es muy básica, por lo que en las futuras actualizaciones se trabajaría en este sistema tan importante para el avance de los juegos.

De esta misma manera, se ha propuesto poder crear una especie de consola virtual para los retos que se puedan ofrecer, de manera que se pueda escribir el código in-situ, se puede probar si lo que se está intentando resolver es del todo correcto, sin tener que hacer uso de programas externos para programarlo y más tarde volver para comprobarlo.

Otra actualización que se ha de realizar, es poder dar salida al juego en modo multiplataforma, que en este caso se encaminaría mucho más a la posibilidad de jugar con él en cualquier soporte digital, como son el caso de los Smartphones (con sistemas operativos Android, IOS, y posiblemente Windows Phone), así como en tabletas digitales y reproductores de música. La idea se crea puesto que actualmente la mitad de la gente hace un mayor uso de este tipo de terminales, y como en la mayoría de los casos, la gente en viajes de larga duración hace un mayor uso de ellos.

Para poder jugar no hace falta estar conectado a internet en esta primera versión, pero para poder ofrecer un mayor plano social, se necesitaría una conexión de datos en los terminales anteriormente citados. Esta idea es bastante insostenible, ya que la gente no querría gastar capacidad de descarga de datos por jugar a un juego educativo, por lo que se proponen dos opciones: juego en linea con otros usuarios, o juego singular, sin necesidad de tener que conectarse.

6.2. Conclusiones.

Después de la explicación, en este último capítulo se expondrá las conclusiones que se han obtenido junto con la finalización del mismo.

En el primer capítulo de este trabajo se marcaron unos objetivos, los cuales eran la base para la creación de esta herramienta. A continuación, se van a repasar esos conceptos, junto con las experiencias en cuanto al grado de satisfacción de la herramienta final.

1. **El objetivo principal del proyecto es facilitar el aprendizaje de una asignatura, relacionada con la programación desarrollada en lenguaje Java, que principalmente no se ha visto anteriormente, y por tanto se intenta ayudar en el primer contacto con la misma.** En este aspecto se mantiene la idea inicialmente mencionada. No se han introducido conceptos relativos a las anteriores asignaturas, y tampoco se profundiza e exceso en los temas tratados sobre la Programación Orientada a Objetos.

2. **Enseñar desde un punto distinto al que se puede dar durante las clases, para que el alumno pueda ver como se pueden hacer las cosas de varias maneras, ya que la programación tiene varios caminos de pensamiento y por tanto distintas posibilidades de hacer un mismo programa con la misma finalidad.** Realmente durante la creación del juego, se ha tenido que trabajar de distinta manera con el lenguaje JavaScript, pero lo que el usuario final percibe no es esto.

3. **Demostrar las salidas que pueden tener los que en algún futuro se dediquen a la programación profesionalmente, creando videojuegos o aplicaciones para cualquier entorno multiplataforma.** Con la creación de esta herramienta queda demostrada la salida que puede ofrecer dedicarse plenamente al mundo de la programación e inclusive al mundo de la animación, aunque en este trabajo no se trate dicho tema

4. **Enseñar que con bajo presupuesto se puede hacer un juego educativo, y que con sólo unas nociones básicas de programación, se realizan proyectos básicos con buena presencia con vistas a un público general.** Este apartado no es del todo cierto, durante la creación del juego, la programación básica estudiada en la carrera ha servido únicamente de base frente a los complejos algoritmos de conectividad entre los distintos personajes alojados en la escena de la Lujuria. Por lo que se puede concluir, con que es necesario dedicarle bastante tiempo para poder realizar una herramienta visualmente agradable y con una funcionalidad que sea un mínimo dentro de los juegos actuales en el mercado.

5. **Sentar las bases, para que en otro proyecto, se pueda crear la función multijugador, en la cual los alumnos se podrán conectar mediante una clave de usuario y una contraseña, y puedan explorar en mucha más medida las capacidades que otorga este juego.** Esto se ha delegado especialmente en la escena de la Lujuria, donde se han creado la mayor parte de las funciones principales, correspondiente a la interacción del jugador con el resto de personajes.

Como se puede observar, el último punto de los tratados es más bien una mejora, y no tanto un objetivo propuesto dentro del mismo proyecto. De tal manera que sirve para que otro estudiante, pueda con su TFG(recordar que el plan antiguo ha desaparecido) dotarle de mejoras en cuanto a lo que se ha expuesto.

PRESUPUESTO

El presupuesto del proyecto está constituido por las distintas parte que en el se divide:

1.Costes asociados de personal.

Los costes que aquí se representan, son los que se establecen en el *Convenio colectivo nacional de empresas de ingeniería y oficinas de estudios técnicos,* y basándose en las últimas tablas salariales correspondientes al año 2011 publicadas en el BOE por el Ministerio de Trabajo e Inmigración [*http://www.boe.es/boe/dias/2012/03/28/pdfs/BOE-A-2012-4284.pdf*], se declara que:

- Coste salarial anual para Nivel 1 (Licenciados y titulados): 23430.82 €.
- Coste de la hora normal trabajada (asumiendo 1806 horas anuales): 12.97€.

Labor	Días totales	Horas / día	Horas totales	Importe
Estudios iniciales	30	4	120	1556.40€
Diseño de la aplicación	40	4	160	2075.20€
Implementación	140	4	560	7263.20€
Documentación	60	4	240	3112.80€
TOTAL	270			14007.60€

Tabla 1. Desglose de costes de acuerdo con tiempo empleado.

2.Costes por materiales.

2.1. Software.

- Software propietario. En esta categoría se indican el sistema operativo con el que se ha trabajado, así como las herramientas de desarrollo usadas.

Licencia	Coste total	Coste asociado
iWorks	115€	57.50€
TOTAL		57.50€

Tabla 2. Desglose de costes de software propio.

- Software de libre distribución. En este apartado se indican todas las herramientas usadas y que no han supuesto coste alguno para la creación del proyecto. En este proyecto han sido el caso de Safari, Unit 3D, Mixamo, Asset store y Google 3D (perteneciente a Trimble).

2.2. Hardware.

Se exponen los distintos equipos usados para la correcta realización del proyecto.

Concepto	Coste total	Coste asociado
MacBook Pro	2500€	500€
Soporte ventilador	15€	5€
TOTAL		505€

Tabla 3. Desglose de costes de hardware.

2.3. Otros conceptos.

Aquí se engloban las partidas que no pertenecen a ninguna de las categorías anteriormente expuestas.

Concepto	Coste total	Coste asociado
Conexión Internet ADSL	400€	40€
Documentación	100€	75€
TOTAL		115€

Tabla 4. Desglose de costes de materiales encuadrados en "Otros".

3. Presupuesto total del proyecto.

Considerando todos los recursos humanos empleados contabilizados, con sus costes asociados, así como el tipo de material empleado, dividiéndolo en hardware y software, el presupuesto necesario para la realización de este proyecto, se corresponde a:

Licencias	Coste asociado
Costes de personal	14007.60€
Costes materiales	677.50€
TOTAL	14685.10€

Tabla 4. Presupuesto total del proyecto.

El presupuesto total de este proyecto, asciende a CATORCE MIL SEISCIENTOS OCHENTA Y CINCO CON DIEZ CÉNTIMOS.

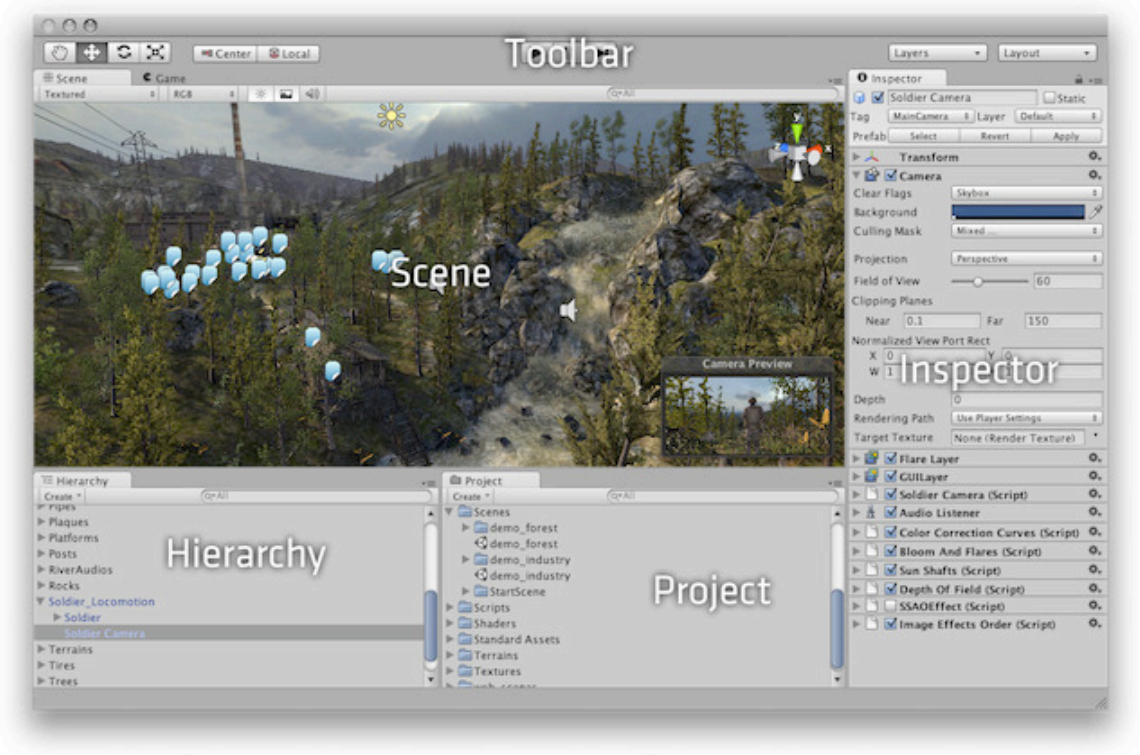
Fdo:

David Gómez Duran. Ingeniero Técnico de Telecomunicaciones.

ANEXO I:

GUÍA BÁSICA DE UNITY3D.

Unyty3D es una empresa dedicada a la infraestructura básica de la creación de videojuegos, ofreciendo sus servicios tanto de forma gratuita (en la que está basado este proyecto), tanto como en la forma de pago, para cualquier plataforma (Mac OS, Windows, e incluso aplicaciones para los teléfonos de nueva generación, los Smartphones).



La pantalla principal del programa se subdivide en varios apartados: Toolbar (barra de herramientas), Inspector(propiedades de los objetos, la escena, etc.), Scene(escena con la que se está trabajando en el proyecto), Project(donde están guardados los recursos de los que dispone el sistema, o se hayan instalado con anterioridad) y Hierarchy(lugar donde se guarda toda la información del actual proyecto, y donde se ejecutan las distintas aplicaciones o atributos asignados a los objetos de la escena).

TOOLBAR

La barra de herramientas se compone de cinco controles básicos, lo cual cada uno de ellos incide de forma determinada y distinta en la zona de edición, que es la escena.

Herramienta de transformación, su utilidad se basa fundamentalmente, en el movimiento tanto de los objetos como de las escenas enteras, pudiendo: arrastrarlas, desplazarlas, girarlas o variar su tamaño.

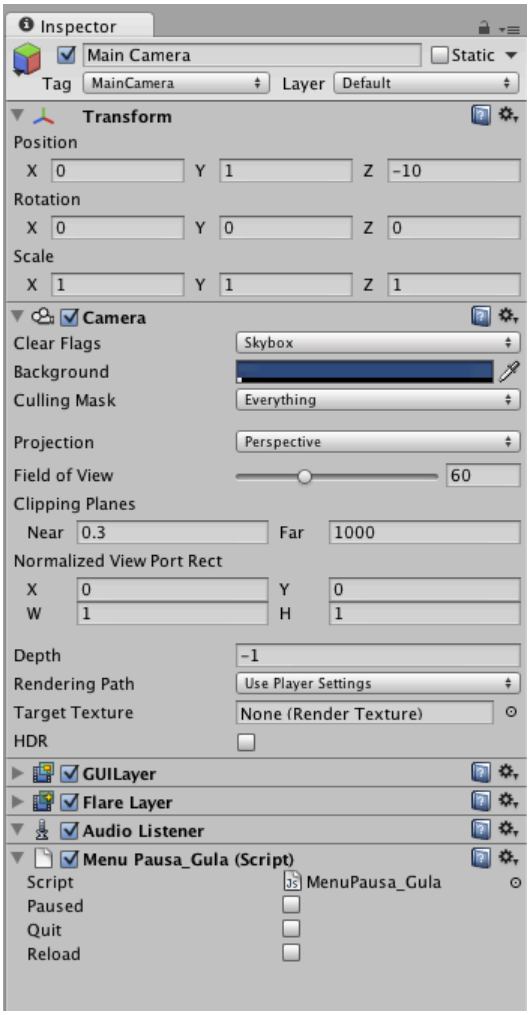
Transformación alterna de la pantalla de visualización.

Play/Pause/Stop durante la ejecución, son usados para una vez creadas las escenas de visualización del juego, poder probar de que estado están los objetos, como se ejecutan los scripts, o simplemente para ver el desarrollo hasta el momento del proyecto.

Controla que objetos se pueden ver en la escena del juego.

Cambia el aspecto de la ventana de trabajo, adoptando distintas posibilidades.

INSPECTOR



Los juegos creados con Unity, contienen información sobre: scripts, sonido, estado de colisión, texturas, colores, y un sin fin de posibles formas.

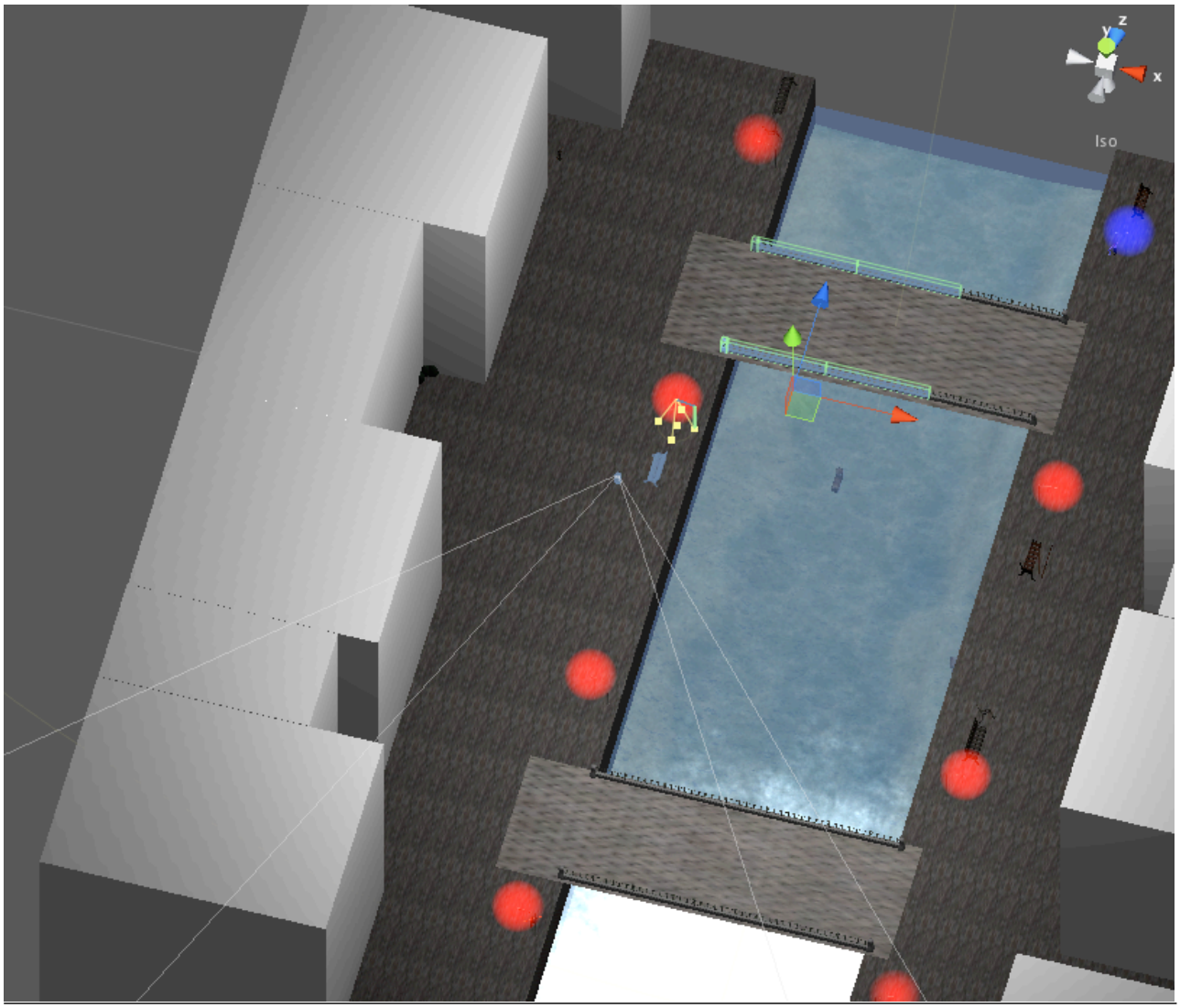
El Inspector es el que nos proporciona todo ese tipo de información asociada al objeto incluido en la escena, así como sus propiedades, y sus componentes.

Es de vital importancia saber controlar este tipo de acceso, ya que con él, se define el tipo de objeto que aparecerá en escena.

Cuando el proyecto está en plena ejecución, es posible saber como afectarían los cambios realizados sobre el objeto, trabajando directamente sobre el Inspector, pero al cerrar la ejecución, todos los cambios realizados, no se verán reflejados, sino que volverán a su estado inicial, antes de haberlos cambiado.

SCENE

La escena es nuestro mundo virtual, en el se crea, se modifica y se reproduce lo que podamos imaginar.



En este tipo de visión, se pueden manejar los objetos con las herramientas explicadas anteriormente. De esta manera, la pronta y rápida familiarización con esta escena, es dentro de todo lo más importante, pues dependiendo de lo que aquí se exponga, saldrá el resultado del proyecto final.

El funcionamiento es el siguiente:

- Si se deja pulsado el botón izquierdo del ratón, se crea un recuadro que selecciona todo lo que dentro de él se encuentre.
- Alt + botón izquierdo del ratón se desplaza el mapa en el que se está.
- Alt + botón central del ratón, sobre cualquier parte de la escena, nos centra donde está puesto el cursor del ratón.
- Alt + botón derecho del ratón, se gira la escena sobre si mismo, idem para sólo presionar el botó izquierdo del ratón.
- Rueda central del ratón, acerca o aleja la visión de la escena.

La vista del juego, será proporcionada desde la posición de la cámara introducida en la escena, y este será el aspecto final de la escena del juego. Y para poder ejecutar esta visión, es necesario pulsar Play, en la pantalla superior central.

Una vez dentro del juego, y habiendo incluido un personaje, de la lista proporcionada por Unity inicialmente, se podrá desplazar por el terreno creado, siendo los controles de movimiento:

- A / cursor izquierdo, desplazamiento lateral izquierdo.
- D / cursor derecho, desplazamiento lateral derecho.
- W / cursor superior, desplazamiento hacia delante.
- S / cursor inferior central, desplazamiento hacia detrás.
- Mouse, visión del personaje, y movimiento del mismo.
- Barra espaciadora, para que el personaje salte en la escena.
- Tecla Shift, manteniéndola, para correr.

Así pues, esta es una guía básica para todo aquel que quiere iniciarse en la creación de cualquier tipo de videojuego, ya sean educativos, como es este caso o, por el contrario, de aventura, estrategia, acción, coches, etc [35].

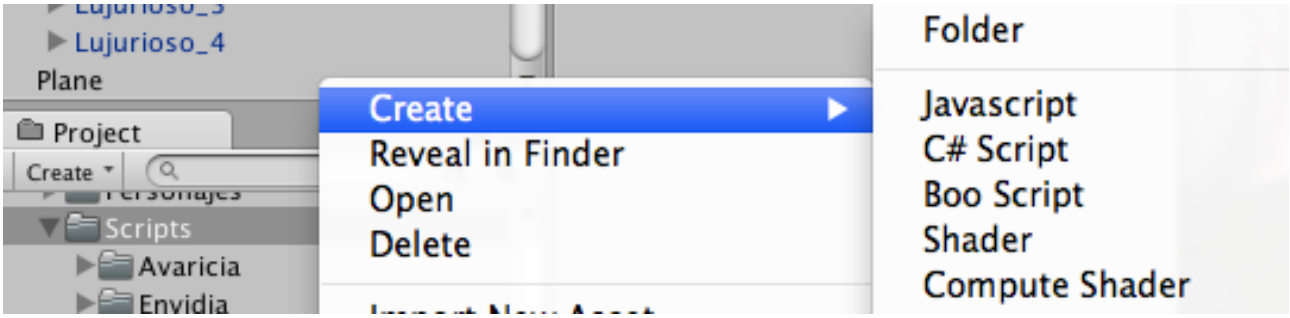
ANEXO II:

GUÍA BÁSICA SOBRE ALGUNAS FUNCIONES DE JAVASCRIPT EN MONODEVELOP.

Como se ha indicado en los capítulos anteriores, Unity ofrece un compilador propio. Este compilador acepta distintos lenguajes, todos ellos para poder crear escenas más dinámicas y reales.

En este nexo, lo que se pretende es indicar muy brevemente como se ha trabajado con el programa a modo de tutorial. Se mostrará como se han de crear los archivos **.js* (ya que con ellos está basada toda la programación del juego) y como algunas funciones ya predeterminadas, incluidas en el compilador de Unity, han permitido poder realizar ciertas acciones más vistosas para el usuario.

Para empezar, es necesario crearse un archivo **.js*. Para poder hacerlo es recomendable crease dentro de la contenedora *Project*, una carpeta donde poder ir guardando todos los scripts que se vayan a usar, más que nada para poder tener un orden y un acceso rápido en caso de necesitar hacer mejoras posteriormente.



Una vez creada la carpeta, se pulsa el botón derecho del ratón para poder acceder a las múltiples acciones ofrecidas por Unity. Dentro del sub-menú seleccionar la opción de JavaScript para poder empezar a realizar la creación del código.

Creado el archivo, pinchar dos veces sobre él, y el mismo programa nos abre la ventana del compilador MonoDevelop. Las herramientas más importantes, para un uso de principiante, son la de poder ejecutar los programas. Es un programa similar al usado en los primeros años de carrera, *Eclipse*.

Estos botones son las principales acciones controladas en la creación del programa. La primera es una ejecución parcial del código con el que se está trabajando, y la segunda es una ejecución total, de todos los programas creados.

La declaración de atributos, es muy distinta en cuanto a la forma de hacerlo en Java.

```
static var isEnter: boolean = false;
```

```
public var Dialogs : int = 0; //Para decir que texto mostrar.
private var iDialogueCnt:int = 0;
private var aDialogs = new Array();

var opcion_1;
var opcion_2;

public var lujurioso1: Texture2D;
public var dante: Texture2D;
```

Para poder crear variables, sin establecer un tipo básico a los atributos, se crea anteponiendo la palabra "*var*" al atributo en cuestión, siendo accesible únicamente por el programa en el que aparece. Una de las funciones más importantes ofrecidas por Unity junto con el compilador, es la posibilidad de ver como varían los valores de los atributos según avanza el juego. De esta manera, se puede acceder al valor que tiene *Dialogs* en cualquier momento durante la ejecución del programa, anteponiéndole la palabra "*public*", de esta manera se podrá ver la información aportada por el atributo en la pestaña *Inspector*, dentro de Unity. En este último caso, se ha inicializado a un tipo básico "*int*", con valor cero.

En caso de querer dotar a un personaje de una imagen dentro de las ventanas de los diálogos(posteriormente se verá como trabajan las GUI), es obligatorio declararlo "*public*", para poder acceder desde el inspector del proyecto y asignar la imagen oportuna, e inicializarlo con la palabra prefijada por el programa *Texture2D*.

En el caso de que se quiera acceder desde otro programa a un atributo creado fuera del mismo, es necesario darle la propiedad de "static", de tal manera que cuando se llame a ese atributo desde otro programa se cambie el valor definido con anterioridad.

```
Texto_Angel_Lujurioso.isEnter = true;
```

De esta manera cada vez que se evoca el *isEnter*, asociado al personaje correspondiente, cambiará de estado, al que se le haya indicado con la referencia a ese atributo.

Una vez mostradas algunas particularidades de los atributos dentro de JavaScript, se pueden mostrar unos de los elementos más usados en cualquiera de las escenas del juego, los GUI.

Los GUI, son la Interfaz Gráfica de Usuario (del inglés Graphycal User Interface), y son los encargados de hacer uso de ciertas herramientas para poder representar la información contenida y mostrarla al usuario final de una manera más comunicativa, y no tan brusca como sería con el código sólo. Dentro de este caso, se han hecho uso de varios comandos pre-definidos, para poder mostrar por pantalla los distintos mensajes. Destacar que todos los bocadillos que aparecen en las escenas, han de hacerlo en una zona determinada de la pantalla, por lo que aquí se muestra cual sería la forma correcta de saber como ponerlos, estableciendo como punto 0, el centro de la pantalla:

```
//Rect(posicion el el ancho de la pantalla,posicion en el alto de la pantalla,ancho del rectangulo,alto del rectandgulo)
```

- GUI.Box. Es el más usado en este trabajo, ya que con él se pueden crear los bocadillos de diálogo en donde aparecen las frases de los personajes. Este GUI, permite presentar el más básico de los bocadillos, siendo únicamente de lectura.

```
if(GUI.Button(Rect(Screen.width/2-200, Screen.height/2+250, 450, 30), "Siguiente")){
```

- GUI.Button. Usado para realizar la función de "si se pincha aquí, sucederá algo", se ve un mayor uso de este código dentro de la escena Lujuria, y las conversaciones de respuesta múltiple.

```
if((GUI.Button(Rect(Screen.width /2 - 200,Screen.height /2 + 250, 400, 30),GUIContent(aDialogs[3], dante)))){
```

- GUI.TextField. Es el menos usado dentro de los GUI, y es el responsable de crear un espacio

donde el usuario puede escribir las contraseñas para poder acceder a la escena desde la página de carga.

```
GUI.TextField (Rect (Screen.width /2 - 150,Screen.height /2 + 100, 300, 47),respuesta, 20);
```

- GUIText. Función interna disponible dentro de Unity, que permite la aparición de mensajes automáticos en la escena cuando así se le indique, se hace uso cuando se intenta hablar con algún personaje de la Lujuria y no se dispone del token apropiado.

En la primera versión que se trabajó con este código, no era necesario incluir al final de cada sentencia el *GUIContent*, el cual permite mostrar el diálogo junto con una pequeña imagen. Posteriormente se tuvo que introducir este aspecto, ya que el programa no compilaba y daba problemas de ejecución, y visualmente era más agradable para saber quien contesta y quien es el que habla, dependiendo de las frases incluidas.

Todos los GUI's que se incluyen en cualquier código, es obligatoriamente hacerlo dentro de una función determinada, llamada:

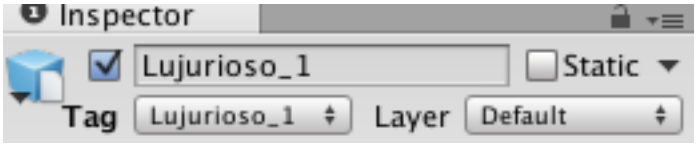
```
function OnGUI(){
```

Cualquier sentencia que contenga dentro de la misma una llamada a cualquier GUI, y no sea incluida dentro de esta función, da error de compilación y no permite la ejecución del programa correctamente.

Otra herramienta que ha resultado muy útil a la hora de obligar al personaje principal a reconocer con que objeto está actuando(es necesario dotar al objeto en cuestión de la funcionalidad *gameObject*) y que es lo que tiene que hacer cuando esto sucede. Esto se hace trabajando desde el *Inspector* dentro de Unity.

1. Primero es necesario renombrar con la palabra deseada el Tag (una etiqueta) asociada al personaje en cuestión. Para cada objeto representativo de la escena, se crea una etiqueta específica, de tal manera que cada uno de ellos tenga nombre propio frente a la programación que se va a crear.

2. Posteriormente se observa dentro de la etiqueta asociada a cada objeto, que correctamente ha quedado nombrado con el nombre deseado.



3. Finalmente, para poder hacer una llamada a dicho objeto dentro del código, es necesario denotarlo con la expresión específica. De esta manera sólo se activa la acción determinada con el personaje impuesto por el código, que anteriormente se había creado en Unity.

```
if(hit.collider.gameObject.tag == "Lujurioso_1"){
```

Esta herramienta demuestra la unión que existe entre Unity y su compilador, ya que de esta manera cualquier objeto que se haya creado dentro del juego, y se haya nombrado propiamente, puede ser accedido por el programa haciendo una llamada al objeto en particular [36].

BIBLIOGRAFÍA

[1] Unity3D, Game List, <http://unity3d.com/gallery/made-with-unity/game-list>, [Última consulta, 18 de octubre de 2012].

[2] Flanagan, D. 2007. *JavaScript: La guía definitiva*. ANAYA. Madrid.

[3] Ceballos, Fco. J. 2002. *Java 2: Curso de programación [Páginas 8 - 9]*. RA-MA. Madrid.

[4] Alighieri, D. 1304 - 1321. *Divina Comedia*.

[5] Alice, <http://www.alice.org/index.php>, [Última consulta, 30 de mayo de 2013].

[6] Snake Wrangling for Kids, <http://code.google.com/p/swfk/>, [Última consulta, 30 de mayo de 2013].

[7] Ruby, <http://www.ruby-lang.org/es/>, [Última consulta, 30 de mayo de 2013].

[8] Kodu, <http://fuse.microsoft.com/projects/kodu>, [Última consulta, 30 de mayo de 2013].

[9] Lego MindStorm NXT, <http://mindstorms.lego.com/en-us/default.aspx>, [Última consulta, 30 de mayo de 2013].

[10] Petit Computer, <http://www.petitcomputer.com>, [Última consulta, 30 de mayo de 2013].

[11] Robomind, <http://www.robomind.net/es/>, [Última consulta, 30 de mayo de 2013].

[12] Scratch, <http://scratch.mit.edu>, [Última consulta, 30 de mayo de 2013].

[13] RPG Maker, <http://www.rpgmakerweb.com>, [Última consulta, 30 de mayo de 2013].

[14] Ogre3D, <http://www.ogre3d.org>, [Última consulta, 30 de mayo de 2013].

[15] Panda3D, <http://www.panda3d.org>, [Última consulta, 30 de mayo de 2013].

[16] JMonkeyEngine, <http://jmonkeyengine.org>, [Última consulta, 30 de mayo de 2013].

[17] UDK, <http://www.unrealengine.com/udk/>, [Última consulta, 30 de mayo de 2013].

[18] Gros Salvat, B. Junio de 2000. *"La dimensión de la socieducación de los videojuegos"*. Publicación nº 12 de Edutec. Revista Electrónica de Tecnología Educativa. Edutec. [Consultado el 19 de octubre de 2012].

[19] Pipo, <http://www.pipoclub.com>, [Última consulta, 30 de mayo de 2013].

[20] Brain Training, <http://www.nintendo.es/Juegos/Nintendo-DS/Brain-Training-del-Dr-Kawashima-Cuantos-anios-tiene-tu-cerebro--270627.html>, [Última consulta, 30 de mayo de 2013].

[21] Where in the world is Carmen Sandiego, <http://www.carmensandiego.com/hmh/site/carmen>, [Última consulta, 30 de mayo de 2013].

[22] Math Blaster, <http://www.mathblaster.com>, [Última consulta, 30 de mayo de 2013].

[23] Wii Sport Resort, <http://www.wiisportsresort.com>, [Última consulta, 30 de mayo de 2013].

[24] El profesor Layton, <http://professorlayton.nintendo.com>, [Última consulta, 30 de mayo de 2013].

[25] Play English, <http://es.playstation.com/psp/games/detail/item275177/PlayEnglish>, [Última consulta, 30 de mayo de 2013].

[26] Pokemon, <http://www.pokemon.com/es/>, [Última consulta, 30 de mayo de 2013].

[27] Sid Meier's Civilitation, <http://www.civilization5.com/?lang=es/>, [Última consulta, 30 de mayo de 2013].

[28] Top Gun Hard Lock, <http://www.topgungame.com>, [Última consulta, 30 de mayo de 2013].

[29] Rosser, J. C., Lynch, P. J. Haskamp, L., Gentile, D. A., & Yalif, A. 2007. *The impact of video games in surgical training*. Archives of Surgery, 142, 181-186, en la web: http://drdouglas.org/drdpdfs/Rosser_etal_2007.pdf [Última consulta, 6 de junio de 2013].

[30] Corrales Garcia, Alberto J. 2010. *Aventura gráfica para el aprendizaje de programación: "Dante"*. TFC. Carlos III, Madrid.

[31] Mixamo, <http://www.mixamo.com>, [Última consulta, 30 de mayo de 2013].

[32] Gómez, D. 2011. *Programación de sistemas: Conocimientos básicos para el nuevo alumno*. Trabajo dirigido. Carlos III. Madrid.

[33] Carroll, L. 1865. *Alicia en el país de las maravillas*.

[34] Goscinny, R. y Uderzo, A. 1959. *Las aventuras de Asterix el galo*.

[35] Unity3D, *Learning the interface*, <http://docs.unity3d.com/Documentation/Manual/LearningtheInterface.html>, [Última consulta, 12 de Septiembre de 2012]

[36] Goldstone, W. 2009. *Unity game Development Essentials*. Packtpub. Reino Unido.